Diss. ETH No. 24894

# Selected Topics in Secure Multi-Party Computation

A thesis submitted to attain the degree of

**Doctor of Sciences of ETH Zurich**
(Dr. sc. ETH Zurich)

presented by

**Daniel Tschudi**
MSc ETH Mathematics, ETH Zurich

born on 15 May 1988
citizen of Glarus GL, and Zürich ZH, Switzerland

accepted on the recommendation of

Prof. Dr. Ueli Maurer, examiner
Dr. Martin Hirt, co-examiner
Prof. Dr. Jesper Buus Nielsen, co-examiner

2018

# Acknowledgments

First of all, I would like to thank my advisor Ueli Maurer for giving me the opportunity to do a PhD in his group and explore the fascinating world of cryptography. His abstract perspective on cryptography lead to many interesting discussions.

I owe a great debt of gratitude to Martin Hirt. He always had time for me and taught me in endless, but enlightening discussions how to transform cool ideas into presentable results. His support and patience were a great help in reaching this point. I would also like to thank him for organizing the applied geometry seminar[1].

Sincere thanks go to Jesper Buus Nielsen for kindly agreeing to serve as a co-examiner on my committee.

I thank all my collaborators on projects both in and outside the scope of this thesis – Christian Badertscher, Juan Garay, Chen-Da Liu Zhang, Julian Loss, Marta Mularczyk, Martin Raszyk, and Vassilis Zikas. In particular, I would like to thank Vassilis for his advice on writing papers.

Special thanks go to my long-term office mates Chen-Da Liu Zhang and Sandro Coretti, who had to endure me and were great partners for discussing research and all kinds of other things. I am also grateful to all the other former and current members of the Information Security and Cryptography Group at ETH that have been my colleagues: Peter Gaži, Maria Dubovitskaya, Björn Tackmann, Pavel Raykov, Gian Pietro Farina, Robert Enderlein, Grégory Demay, Christian Matt, Daniel Jost, Gregor Seiler, Christopher Portmann, and Fabio Banfi.

I would also like to thank our group and department secretaries Beate Bernhard, Claudia Günthart, and Denise Spicher for helping with all the

[1]a.k.a. the pool billiards sessions over lunch

administrative tasks.

Doing a PhD can be harsh at times. I would therefore thank my parents Edith and Thomas, and my brothers Christian and Lukas, for their support during such times—and all other times as well.

# Abstract

Secure multi-party computation (MPC) allows a set of $n$ parties to evaluate a function $f$ in the presence of an adversary who corrupts a subset of the parties. In this work we investigate three selected topics in the area of MPC.

Most MPC protocols require that parties are pair-wise connected by means of secure channels. To run an MPC over an incomplete network, secure message transfer protocols (SMTP) can be used. However, classic SMTP leaks information about the topology of the underlying network. In the first part of this thesis, we present the first topology-hiding communication protocol for incomplete networks which makes black-box use of the underlying cryptographic assumptions. The protocol tolerates any adversary who passively corrupts arbitrarily many network nodes. This protocol allows to make any MPC protocol with passive security topology-hiding. We further show how to construct anonymous broadcast without using expensive MPC to setup the original pseudonyms.

Broadcast channels are an important primitive used in many MPC protocols. It is well-known that broadcast channels can be achieved with perfect security if and only if the fraction of active cheaters is less than a third. A natural question initially raised by Lamport, is whether there are weaker, still useful primitives achievable from authenticated channels. In the second part of the thesis we investigate generalizations of the broadcast setting in two directions: weaker forms of consistency guarantees are considered, and other resources than merely bilateral channels are assumed to be available. The ultimate goal of this line of work is to arrive at a complete classification of consistency specifications.

In the third part of the thesis, we consider active, general adversaries which are characterized by a so-called adversary structure $\mathcal{Z}$. The adversary structure enumerates all possible subsets of corrupted parties. Protocols for general adversaries are “efficient” in the sense that they require $|\mathcal{Z}|^{\mathcal{O}(1)}$ bits of communication. However, as $|\mathcal{Z}|$ is usually very large (even exponential in $n$), the exact exponent is very relevant. For the setting with perfect security we present a protocol which requires $\mathcal{O}(|\mathcal{Z}|^2)$ bits of communication. For the setting with statistical security we present a protocol which requires $\mathcal{O}(|\mathcal{Z}|)$ bits of communication. Both protocols have a better communication complexity than previous protocols.

# Zusammenfassung

Sichere Mehrspielerberechnung (MSB) erlaubt $n$ Parteien eine Funktion $f$ in Präsenz eines Gegner, der eine Teilmenge der Parteien korrumpiert, sicher zu berechnen. In dieser Arbeit untersuchen wir drei ausgesuchte Themenbereiche im Gebiet der MSB.

Die meisten MSB-Protokolle erfordern, dass die Parteien paarweise durch sichere Kanäle verbunden sind. Zur Durchführung einer MSB über einem unvollständigen Netzwerk können sichere Nachrichtenübertragungsprotokolle (SNÜP) verwendet werden. Klassische SNÜP geben jedoch Informationen über die Topologie des zugrunde liegenden Netzwerks preis.

Im ersten Teil dieser Dissertation wird das erste Topologie-verbergende SNÜP für unvollständige Netzwerke vorgestellt, das die zugrundeliegenden kryptographischen Annahmen in “Blackbox” umsetzt. Das Protokoll toleriert Gegner, die beliebig viele Netzwerkknoten passiv korrumpieren können. Dieses Protokoll ermöglicht es, jedes passiv sichere MSB-Protokoll auch Topologie-verbergend zu machen. Zusätzlich wird gezeigt, wie anonymer Rundfunk konstruiert werden kann, ohne dass teure MSB für die Einrichtung der Pseudonyme verwendet werden muss.

Rundfunkkanäle sind ein wichtiges Grundelement, das in vielen MSB-Protokollen verwendet wird. Es ist wohlbekannt, dass Rundfunkkanäle genau dann konstruiert werden können, wenn der Anteil an aktiven Betrügern weniger als ein Drittel beträgt. Eine natürliche Frage, die zuerst von Lamport aufgeworfen wurde, ist, ob es schwächere, aber noch immer nützliche Primitiven gibt, die aus authentifizierten Kanälen erreichbar sind. Im zweiten Teil der Dissertation werden zwei Richtungen zur Verallgemeinerungen dieses Unmöglichkeitsresultats untersucht: Schwächere Formen von Konsistenzgarantien werden betrachtet und andere Ressourcen als nur bilaterale Kanäle werden als verfügbar vorausgesetzt. Das ultimative Ziel in dieser Fragestellung ist, eine vollständige Klassifizierung der Konsistenzspezifikationen zu erreichen.

Im dritten Teil der Dissertation werden aktive Allgemeingegner betrachtet, die durch eine sogenannte Gegnerstruktur $\mathcal{Z}$ gekennzeichnet sind. Die Gegnerstruktur zählt alle möglichen Mengen korrumpierter Parteien auf. Protokolle für Allgemeingegner sind "effizient" in dem Sinne, dass sie $|\mathcal{Z}|^{\mathcal{O}(1)}$ Bits Kommunikation benötigen. Da $|\mathcal{Z}|$ normalerweise sehr gross ist (sogar exponentiell in $n$), ist der exakte Exponent äusserst relevant. Für perfekte Sicherheit wird ein Protokoll präsentiert, das $\mathcal{O}(|\mathcal{Z}|^2)$ Bits Kommunikation benötigt. Für statistische Sicherheit wird ein Protokoll präsentiert, das $\mathcal{O}(|\mathcal{Z}|)$ Bits Kommunikation benötigt. Beide Protokolle haben eine bessere Kommunikationkomplexität als alle bisherigen Protokolle.

# Contents

# Chapter 1

# Introduction

> Alice, Bob, Charlie and Eve want to listen to some music. Bob suggests that everyone discloses their favorite artists such that they can choose music which is fine for everyone. However, from past experience Charlie and Eve both distrust each other claiming that the other person will for sure ridicule them for their music taste. Unfortunately, their friend Trent whom they all trust is sick and not there to help and choose the music for them. Alice, eager to listen to some music, interrupts the ensuing discussion and starts with "Have you ever heard of MPC?"

The above problem of choosing music is a special case of private set intersection. Private set intersection is one of a large class of problems where a set of mutually distrusting parties would like to do a joint computation. If a trusted third-party is available such problems can be solved by outsourcing the computation to the trusted party. If no trusted party is available the problems can be solved using a secure multi-party computation (MPC) protocol. More generally, MPC protocols enable the parties to emulate a trusted party which can do an arbitrary computation on the parties' (secret) inputs.

There are many MPC problems known in the literature. In Chaum's spymaster problem two spymasters want to learn the names of double agents, i.e. agents which work for both spy agencies. For obvious reasons, both spymasters are reluctant to give the other the list of all their agents. In Yao's millionaires problem, two millionaires want to determine who is richer without revealing anything else about their fortunes. In both cases MPC allows to compute the desired information without revealing any other information.

More common applications of MPC include online auctions, e-voting or the protection of customer data.

## 1.1 Secure Multi-Party Computation

In this section we give a short introduction to *secure multi-party computation* and provide an overview of the different models used. For a more detailed introduction see, e.g, [CDN15].

### 1.1.1 Multi-Party Computation Models

**Parties and Computation.** In the secure multi-party computation problem a set of $n$ parties $\mathcal{P} = \{P_1, ..., P_n\}$ (also known as players or processors) wants to compute a function $f$ over some finite field $\mathbb{F}$. The function is normally specified by a circuit $\mathcal{C}$ consisting of input, output, random, addition, and multiplication gates. In an ideal world, a trusted party does all the computation. In the real world, parties emulate the trusted party by using some MPC protocol $\Pi$.

**Communication Model.** The parties are connected over a network of pairwise *channels*. Additionally parties may have access to *broadcast channels*. A broadcast channel allows a party (the sender) to consistently distribute a message among all other parties. The network (topology) of channels can be *complete* or *incomplete*.

The assumed (broadcast) channels can be *insecure*, *authentic*, or even *secret* (authentic and private).

In this work we assume a *synchronous* communication model, where all channels have a known and constant upper-bound on the delay and parties have a common clock. This allows for *round-based* protocols.

**Adversary Model.** Dishonesty is modeled in terms of a *central adversary* $\mathcal{A}$ who corrupts parties. There are three different corruption models:

During the execution of the protocol *passively or semi-honestly* corrupted parties follow protocol instructions but the adversary can access their internal state. A *fail-stop* corrupted party follows the protocol instructions until the adversary decided to crash it; from then on the party does no longer participate in the protocol. In particular, it does not send out messages to parties. The adversary can access the internal state of *actively* corrupted parties and make them deviate from the protocol at will. Parties which are not corrupted are called *honest*.

Often, a *threshold adversary* is assumed which is characterized by an upper bound $t$ on the number of corrupted parties. Threshold adversaries are a special case of *general adversaries*. The corruption choice of a general adversary is limited by means of an adversary structure $\mathcal{Z} = \{Z_1, \ldots, Z_\ell\} \subseteq 2^{\mathcal{P}}$, i.e., the set $Z*$ of corrupted parties must be an element of $\mathcal{Z}$.

A *static* adversary must specify the set of corrupted parties before the protocol execution. In contrast, an *adaptive* adversary can corrupt parties during the protocol execution.

Finally, we distinguish between *computationally unbounded* and *computationally bounded* adversaries.

**Security Model.** We say a protocol is secure if anything the adversary achieves during the execution of the protocol can as well be achieved in the ideal world where a trusted party does the computation. We distinguish between *information-theoretic* and *cryptographic* security. In information-theoretic security we require that for every adversary in the real world there exists an adversary in the ideal world such that both the information the adversary gets and the output of honest parties are identically distributed for *perfect* security respectively statistically close for *unconditional* security. In cryptographic security the indistinguishability between the real-world and the ideal-world relays on assumed the bounded computing power of the distinguisher and the (assumed) hardness of some computational problems.

**Efficiency of MPC.** There are different measures for the protocol *complexity*. First, we denote by the *round* complexity the (maximal) number of rounds the protocol requires. Next, we denote by the *communication* complexity the number of bits sent or received by honest parties. For the communication complexity we only consider messages which should have been received according to the protocol description. The communication complexity can be split into the communication complexity for normal channels and the *broadcast* complexity which measures the number of broadcast bit. Sometimes, one also considers the *computational* complexity of the (local) computation of an honest party.

### 1.1.2 History of MPC

MPC was introduced by Yao [Yao82]. A first solution (with computational security) was given by Goldreich, Micali, and Wigderson [GMW87]. Later solutions [BGW88, CCD88, RB89] provide statistical and even perfect security. All these protocols consider threshold adversaries This was generalized in [HM00] by considering general adversaries.

In the setting with perfect active security, MPC is achievable if and only if $t < \frac{n}{3}$, respectively $\mathcal{Q}^3(\mathcal{P}, \mathcal{Z})$ (the union of no *three* sets in $\mathcal{Z}$ covers $\mathcal{P}$). In the setting with statistical or cryptographic active security, MPC is achievable if and only if $t < \frac{n}{2}$, respectively $\mathcal{Q}^2(\mathcal{P}, \mathcal{Z})$ (the union of no *two* sets in $\mathcal{Z}$ covers $\mathcal{P}$) if given broadcast.

## 1.2 Contributions

The contributions of this thesis comprise a protocol for topology-hiding communication [HMTZ16], a classification of consistency specifications [LMT16, LMT17] and an efficient protocol for general adversary MPC [HT13].

### 1.2.1 Topology-Hiding Communication

In MPC it is commonly assumed that parties are connected over a complete network of channels. If the network has an incomplete topology, one can use *secure message transmission* (SMT) to build pairwise secure channels between parties and thus complete the network. Since the introduction of SMT in [DDWY90] the problem has been widely studied under the aspect of different settings and security requirements (see for example [SNR04, FFGV07, KS08]).

However, in those work the topology of the underlying communication network is not treated as private data. In fact, SMT protocols require that parties have knowledge of the network topology. The problem of *topology-hiding* secure multi-party computation over an incomplete network was introduced in [MOR15]. In this work a proof-of-concept network-hiding communication protocol is given for the computational setting (i.e., assuming secure public-key encryption) tolerating a passive (semi-honest) and static adversary. At a very high level, [MOR15] uses public-key encryption and (passive) multi-party computation to topology-hidingly emulate a communication network. This network is then used to execute an arbitrary multi-party protocol in which parties communicate over a complete communication network.

In Chapter 3 we present the first topology-hiding communication protocol for incomplete networks which makes black-box use of the underlying cryptographic assumption—in particular, a public-key encryption scheme—and tolerates any adversary who passively corrupts arbitrarily many network nodes. Our solution is based on a new, enhanced variant of threshold homomorphic encryption, in short, TH-PKE, that requires no a-priori setup and allows to circulate an encrypted message over any (unknown) incomplete network and then decrypt it without revealing any network information to intermediate nodes. We show how to realize this enhanced TH-PKE from the DDH assumption. The black-box nature of our scheme, along with some optimization tricks that we employ, makes our communication protocol more efficient than previous solutions.

We then use our communication protocol to make any passively secure MPC protocol topology-hiding with a reasonable—i.e., for simple networks, polynomial with small constants—communication and computation overhead. We further show how to construct anonymous broadcast without using expensive MPCs to setup the original pseudonyms.

### 1.2.2 Classification of Consistency Specification

A classical problem in MPC is to construct a broadcast channel from authenticated channels allowing a sender to send a value consistently to all (honest) receivers. Lamport showed in [Lam83] that this is possible to perfectly construct authenticated broadcast if and only if the fraction of dishonest parties is less than a third.

A natural question, first raised by Lamport, is whether there are weaker, still useful primitives achievable from authenticated channels. He proposed weak broadcast, where the validity condition must hold only if all parties are honest, and showed that it can be achieved with an unbounded number of protocol rounds, while broadcast cannot, suggesting that weak broadcast is in a certain sense weaker than broadcast.

In Chapter 4 we deepen the investigation of the separation between broadcast and authenticated channels. To this end, we use consistency specifications [Mau04] to model the guarantees primitives such as broadcast provide for the outputs of honest parties.

First, we consider a generalized version of so called "scenario"-proofs (see, e.g., [FLM85]). This allows us to prove a stronger impossibility result for three-party broadcast. Namely, even if two of the parties can broadcast, one cannot construct authenticated broadcast for the third party. Next, we present a complete classification of three-party specifications with a binary input and two binary outputs. Finally, we prove a strong separation between authenticated channels and broadcast by exhibiting a new primitive, called XOR-cast, which satisfies two conditions: (1) XOR-cast is strongly unrealizable (even with small error probability) from authenticated channels (which is not true for weak broadcast), and (2) broadcast is strongly unrealizable from XOR-cast (and authenticated channels). This demonstrates that the hierarchy of three-party primitives has a more complex structure than previously known.

### 1.2.3 Efficient General Adversary MPC

In MPC one normally considers threshold adversaries which can corrupt up to $t$ parties. An alternative are general adversaries, characterized by a so-called adversary structure $\mathcal{Z}$ which enumerates all possible subsets of corrupted parties. In particular for small sets of parties general adversaries better capture real-world requirements than classical threshold adversaries.

In Chapter 5 we consider the efficiency of secure MPC against active general adversaries. Protocols for general adversaries are "efficient" in the sense that they require $|\mathcal{Z}|^{\mathcal{O}(1)}$ bits of communication. However, as $|\mathcal{Z}|$ is usually very large (even exponential in $n$), the exact exponent is very relevant. In the setting with perfect security, the most efficient protocol previously known communicates $\mathcal{O}(|\mathcal{Z}|^3)$ bits; we present a protocol for this setting which communicates $\mathcal{O}(|\mathcal{Z}|^2)$ bits. In the setting with statistical security, $\mathcal{O}(|\mathcal{Z}|^3)$ bits of communication is needed in general (whereas for a very restricted subclass of adversary structures, a protocol with communication $\mathcal{O}(|\mathcal{Z}|^2)$ bits is known); we present a protocol for this setting (without limitations) which communicates $\mathcal{O}(|\mathcal{Z}|^1)$ bits.

# Chapter 2

# Preliminaries

In this chapter we introduce some general notation and provide the definition of some fundamental cryptographic primitives and hardness assumptions.

## 2.1 General Notation

We denote by $\mathcal{P} = \{P_1, ..., P_n\}$ a set of $n$ parties. For convenience, we will sometimes write $i$ instead of $P_i$. We distinguish between the subset of honest parties $H \subseteq \mathcal{P}$ and the dishonest parties in the complement $\overline{H} := \mathcal{P} \setminus H$. The (central) adversary is denoted by $\mathcal{A}$.

Throughout this work, the security parameter is denoted $\kappa$. We write $\mathrm{negl}(\kappa)$ to refer to a negligible function of $\kappa$. (See [Gol01] for a formal definition of negligible functions.)

For an algorithm $A$ we write $(y_1, \ldots, y_k) \leftarrow A(x_1, \ldots, x_k)$ to denote that $(y_1, \ldots, y_k)$ are outputs of $A$ given inputs $(x_1, \ldots, x_k)$. For a probabilistic algorithm $B$ we write $(y_1, \ldots, y_k) \leftarrow B(x_1, \ldots, x_k; r)$ where $r$ is the chosen randomness. If we write $(y_1, \ldots, y_k) \twoheadleftarrow B(x_1, \ldots, x_k)$ instead, we assume that the randomness has been chosen uniformly.

Finally, we denote by $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$ the set of natural numbers. For any $n \in \mathbb{N}$ we write $[n]$ for the set $\{1, \ldots, n\}$. Moreover, $\mathbb{F}$ denotes an arbitrary finite field. For a set of values $V$ and a index-set $I$, we denote by $M^I$ the Cartesian product $\bigtimes_{i \in I} M$. A tuple $\vec{v}_I \in V^I$ contains for each index $i \in I$ value $\vec{v}_i$. We denote by $\vec{v}_{I|I'} \in M^{I'}$ the projection of $\vec{v}_I \in M^I$ to entries in $I' \subseteq I$, i.e, the tuple $\vec{v}_{I|I'} \in M^{I'}$ contains exactly the values of $\vec{v}_I \in M^I$ for index set $I'$.

## 2.2 Cryptographic Primitives

### 2.2.1 Broadcast Channel

A *broadcast channel* $\mathrm{BC}_s$ allows the sender $P_s$ to distribute a message $m$ among the parties in $\mathcal{P}$ such that:

**Consistency:** All honest parties output the same message $m'$.

**Validity:** If the sender $P_s$ is honest, then all honest parties output $m' = m$.

More generally, one can define broadcast channels $\mathrm{BC}_{s,R}$ where the sender's message is only guaranteed to be output to the recipient set $R \subseteq \mathcal{P}$. However, in this work the recipient set is generally assumed to be the whole party set $\mathcal{P}$.

### 2.2.2 Public-Key Encryption

A *public-key encryption* (PKE) scheme with public-key space $\mathcal{PK}$, secret-key space $\mathcal{SK}$, message space $\mathcal{M}$, and ciphertext space $\mathcal{C}$ is defined as three algorithms $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ all parameterized by security parameter $\kappa$ where:

1. The (probabilistic) *key-generation* algorithm $\mathsf{KeyGen}$ outputs a public key $\mathtt{pk} \in \mathcal{PK}$ and a secret key $\mathtt{sk} \in \mathcal{SK}$.
2. The (probabilistic) *encryption algorithm* $\mathsf{Enc}$ takes a public key $\mathtt{pk} \in \mathcal{PK}$ and a message $m \in \mathcal{M}$ and outputs a ciphertext $c \leftarrow \mathsf{Enc}(\mathtt{pk}, m; r)$.
3. The *decryption algorithm* $\mathsf{Dec}$ takes a secret key $\mathtt{sk} \in \mathcal{SK}$ and a ciphertext $c \in \mathcal{C}$ and outputs message $m \leftarrow \mathsf{Dec}(\mathtt{sk}, c)$.

A PKE scheme is *correct* if for any key pair $(\mathtt{pk}, \mathtt{sk}) \leftarrow \mathsf{KeyGen}$ and any message $m \in \mathcal{M}$ it holds (with probability 1 over the randomness of $\mathsf{Enc}$) that $m = \mathsf{Dec}(\mathtt{sk}, \mathsf{Enc}(\mathtt{pk}, m; r))$.

**Chosen-Plaintext Security.** *Indistinguishability under chosen-plaintext attacks* (IND-CPA) considers an adversary trying to decide which of two messages of his choice is encrypted in a given ciphertext. More formally, a PKE scheme is IND-CPA secure if an (efficient) adversary has not a non-negligible advantage in winning the following game.

1. The game generates a key pair $(\mathtt{pk}, \mathtt{sk}) \leftarrow \mathsf{KeyGen}$ and chooses a random bit $b$. Then the games sends $\mathtt{pk}$ to the adversary (this allows him to generate encryptions of arbitrary messages).

2. The adversary sends two messages $m_0$ and $m_1$ of the same length and the game returns $c = \mathsf{Enc}(\mathtt{pk}, m_b)$.

3. The adversary sends a bit $b'$. If $b = b'$ the adversary has won the game.

### 2.2.3 Threshold Public-Key Encryption

A $(\ell, \ell)$*-threshold public-key encryption* (T-PKE) scheme with public-key space $\mathcal{PK}$, secret-key space $\mathcal{SK}$, message space $\mathcal{M}$, and ciphertext space $\mathcal{C}$ is defined as four algorithms $\mathsf{TKeyGen}$, $\mathsf{Enc}$, $\mathsf{ShareDecrypt}$, $\mathsf{Decode}$ all parameterized by security parameter $\kappa$ and the *threshold* $\ell$ where:

1. The (probabilistic) key-generation algorithm $\mathsf{TKeyGen}$ outputs a public key $\mathtt{pk} \in \mathcal{PK}$ and a secret key $\mathtt{sk}$ which consists of a vector of sub-keys $\mathtt{sk}_i \in \mathcal{SK}$, i.e., $\mathtt{sk} = (\mathtt{sk}_1, \ldots, \mathtt{sk}_\ell)$.

2. The (probabilistic) *encryption algorithm* $\mathsf{Enc}$ takes a public key $\mathtt{pk} \in \mathcal{PK}$ and a message $m \in \mathcal{M}$ and outputs a ciphertext $c \leftarrow \mathsf{Enc}(\mathtt{pk}, m; r)$.

3. The *decryption-share algorithm* $\mathsf{ShareDecrypt}$ takes a sub-key $\mathtt{sk}_i \in \mathcal{SK}$, and a ciphertext $c \in \mathcal{C}$ as inputs and outputs a decryption share $\mathtt{x}_i \leftarrow \mathsf{ShareDecrypt}(\mathtt{sk}_i, c)$.

4. The *decoding algorithm* $\mathsf{Decode}$ takes decryption shares $\mathtt{x}_1, \ldots, \mathtt{x}_\ell \in \mathcal{DS}$ and a ciphertext $c \in \mathcal{C}$ as inputs and outputs a message $m \leftarrow \mathsf{Decode}(\mathtt{x}_1, \ldots, \mathtt{x}_\ell, c)$.

A T-PKE scheme is *correct* if for any key pair $\texttt{pk}, \texttt{sk} = (\texttt{sk}_1, \ldots, \texttt{sk}_\ell)$ generated using $\mathsf{KeyGen}$ and any message $m \in \mathcal{M}$ it holds (with probability 1 over the randomness of $\mathsf{Enc}$) that $m = \mathsf{Decode}(\texttt{x}_1, \ldots, \texttt{x}_\ell, c)$ where $\texttt{x}_i = \mathsf{ShareDecrypt}(\texttt{sk}_i, c)$ and $c = \mathsf{Enc}(\texttt{pk}, m; r)$.

**Remark.** *More generally, one can define a $(t, \ell)$-threshold public-key encryption scheme where the secret key consists of $\ell$ key shares. To efficiently decrypt a ciphertext at least $t$ out of the $\ell$ key shares are required.*

## 2.3 Hardness Assumptions

### 2.3.1 Decisional Diffie-Hellman Assumption

Let $\langle G, \cdot \rangle$ be a cyclic group of prime order $q$ and let $g$ be a generator of $G$. The *decisional Diffie–Hellman* (DDH) problem for $G$ is the problem of distinguishing for given group elements $(\alpha, \beta, \gamma)$ whether they are independent uniform random in $G$ or whether $\alpha = g^a$ and $\beta = g^b$ are independent uniform random and $\gamma = g^{ab}$.

The DDH assumption holds in $G$ if it is computationally hard to solve the DDH problem in $G$.

# Chapter 3

# Topology-Hiding Communication

The content of this chapter is based on [HMTZ16].

## 3.1 Introduction

Secure communication is perhaps the central goal of cryptography. It allows a sender, Alice, to securely transmit a message to a receiver, Bob, so that even if some eavesdropper, Eve, is intercepting their communication she cannot figure out anything about the transmitted message. When Alice and Bob share a physical (but potentially tappable) communication channel, this task can be easily carried out by use of standard public-key cryptography techniques, e.g., Bob sends to Alice his public key who uses it to encrypt her message and send it over the physical communication channel to Bob. But this idealized scenario occurs rarely in modern networks, such as the Internet, where Alice and Bob would most likely not share a physical channel and would, instead, have to communicate over some (potentially incomplete) network of routers. Without further restrictions, the above modification marginally complicates the problem as it can be directly solved by means of a private flooding scheme. In such a scheme, Alice encrypts her message, as before, and sends it to all her immediate neighbors, i.e., network routers with which she shares physical links, who then forward it to their immediate neighbors, and so on, until it reaches Bob. Clearly, if Alice has a path to Bob and the forwarding step is repeated as many times as the length of this path, the message will reach Bob. And the fact that the intermediate routers only see encryptions of the transmitted message means that they do not learn anything about the message.

But modern distributed protocols often require much more than just privacy of the transmitted message. For example, ensuring anonymity in communication is a major goal of security as it, for example, protects against censorship or coercion. Similarly, as privacy awareness in social networks increases, users might not be willing to reveal information about the structure of their peer graph (i.e., their Facebook friends graph) to outsiders. Other applications might require to hide a communicating agent's location, as is the case in espionage or when using mobile agents to propagate information through some ad-hoc network, e.g., in vehicle-to-vehicle communication. All these applications require a routing scheme, that hides the topology of the underlying communication network. Evidently, using the simple private flooding strategy does not hide the topology of the underlying communication network as, for example, an eavesdropping router can easily determine its distance (and direction) to the sender by observing in which round (and from whom) it receives the first encryption.

### 3.1.1 Related Literature

The problem of routing through an incomplete network has received a lot of attention in communication networks with a vast amount of works aiming at optimizing communication complexity in various network types. In the following, however, we focus on the cryptographic literature which is more relevant to our goals—namely network hiding communication—and treatment.

Perhaps the main venue of work in which keeping the network hidden is a concern is the literature on anonymous communication, e.g., [Cha03, RR98, SGR97]. These works aim to hide the identity of the sender and receiver in a message transmission, in a way that protects these identities even against traffic analysis. In a different line of work initiated by Chaum [Cha81], so called *mix* servers are used as proxies which shuffle messages sent between various peers to disable an eavesdropper from following a message's path. This technique has been extensively studied and is the basis of several practical anonymization tools. An instance of the mix technique is the so called onion routing [SGR97, RR98], which is perhaps the most wide-spread anonymization technique. Roughly, it consists of the sender applying multiple encryptions in layers on his message, which are then "peeled-off" as the cipher-text travels through a network of onion routers towards its destination. An alternative anonymity technique by Chaum [Cha88] and implemented in various instances (e.g.,[Bd90, GJ04, GGOR14]) is known as *Dining Cryptographers networks*, in short DC-nets. Here, the parties themselves are responsible for ensuring anonymity.

The question of hiding the communication network was also recently addressed in the context of secure multi-party computation by Chandran *et al.* [CCG+15]. This work aims at allowing $n$ parties to compute an arbitrary given function in the presence of an adaptive adversary, where each party communicates with a small (sublinear in the total number of parties) number of its neighbors. Towards this goal, [CCG+15] assumes that parties are secretly given a set of neighbors that they can communicate with. Because the adversary is adaptive, it is crucial in their protocol that the communication does not reveal much information about the network topology, as such information would allow the adversary to potentially discover the neighbors of some honest party, corrupt them, and isolate this party, thereby breaking its security. Another work which considers such an adaptive corruption setting is the work of King and Saia [KS10], which is tailored to the Byzantine agreement problem. We note in passing that the result of [CCG+15, KS10] was preceded by several works which considered the problem of MPC over incomplete networks. However, these works do not aim to keep the network hidden as they either only consider a static adversary, [1] e.g., [BGT13], and/or they only achieve so called *almost everywhere computation* [GO08, KSSV06a, KSSV06b, CGO15] where the adversary is allowed to isolate a small number of honest parties.

In our work we improve the recent work of Moran, Orlov, and Richelson [MOR15], which considers the problem of *topology-hiding secure multi-party computation* over an incomplete network in the computational setting (i.e., assuming secure public-key encryption) tolerating a passive

[1] Learning the network topology through the communication cannot help a static adversary to isolate any honest party as the set of corrupted parties is fixed before the execution of the protocol.

(semi-honest) and static adversary. At a very high level, [MOR15] uses public-key encryption and (passive) multi-party computation to implement a proof-of-concept network-hiding communication protocol, which emulates a complete network of secure channels. This emulated network is then used to execute an arbitrary multi-party protocol in which parties communicate over a complete communication network, e.g., [GMW87, Pas04]. In fact, as noted in [MOR15], relying on a computational assumption seems inevitable, as in the information-theoretic setting the work of Hinkelmann and Jakoby [HJ07] excludes fully topology-hiding communication.[2] As [MOR15] was the starting point for our work, we include a detailed comparison of our results with [MOR15] in Section 3.1.3.

### 3.1.2 Contributions

In this work we present the first network-hiding communication protocol which makes black-box use of public-key encryption and, for networks with reasonable degree and diameter, has a moderate communication and computation complexity. Our protocol allows the parties to communicate over an incomplete network of point-to-point channels in a way which computationally hides both the transmitted message and the neighborhood of honest parties from an adversary *passively* corrupting arbitrary many parties. We remark that as pointed out in [CCG$^+$15], when the communication graph is to be kept hidden, the adversary cannot be eavesdropping on communication channels, and in particular cannot be informed when a message is transmitted over some channel. We resolve this issue by assuming, along the lines of [MOR15], a special network functionality (cf. Section 3.2).

A bit more concretely, the high-level idea of our construction is to enhance the naïve private flooding-protocol by using homomorphic public-key encryption (in short, PKE). The starting point of our approach is the observation—underlying also the construction from [MOR15]—that the flooding protocol would be topology-hiding if the parties could not read intermediate messages. But instead of using, as in [MOR15], expensive nested MPCs for ensuring this fact (see below for a high-

[2]To our understanding the result of [HJ07] does not apply to the case where a strong information-theoretic setup, e.g., sufficiently long correlated randomness, is available to the parties. Extending this results to that setting is an interesting open problem.

level description of [MOR15]) we use a version of threshold PKE with additional homomorphic properties. We also show how to implement our enhanced threshold PKE definition assuming hardness of the Decisional Diffie-Hellmann (DDH) problem.

To demonstrate our ideas, imagine there was a world in which parties (corresponding to all intermediate routers) could encrypt with a homomorphic public-key encryption scheme where the private (decryption) key is known to nobody, but instead parties have access to a decryption oracle. Provided that the associated PKE-scheme is semantically secure, parties can enhance the flooding protocol as follows: Alice encrypts its message and starts the flooding; in each step of the flooding protocol, the intermediate party—which, recall, is supposed to forward the received ciphertext—first re-randomizes the ciphertext and then forwards it. Once the message arrives to Bob, he invokes the decryption oracle to open its final ciphertext. We observe that in this case the adversary does no longer learn anything from intermediate messages, the protocol is thus topology-hiding.

There are two major challenges with the above approach. First, if intermediate parties are silent until a message reaches them during the flooding, then the adversary observing this fact can use it to deduce information about the network. E.g., if a neighbor $P_i$ of a corrupted party has not sent anything by the second round of the flooding protocol, then the adversary can deduce that $P_i$ is not a neighbor of Alice. Secondly, we need a way to implement the decryption oracle. Observe that using an off-the-shelf threshold decryption scheme and have decryption shares exchanged by means of flooding would trivially destroy the topology-hiding property; and the same is the case if we would use an MPC protocol for this purpose, unless the MPC were itself topology-hiding. In the following we discuss how we solve each of the protocols, separately.

The first issue—information leakage from silent parties—can be solved by having every party send messages in every round. As simple as this idea might seem, it has several difficulties. For starters, the messages that are injected by intermediary parties should be indistinguishable from encryptions, as otherwise adding this noise makes no difference. But now, there is a new issue that the intermediate parties cannot tell which of the indistinguishable messages they receive contains the initial message sent by Alice. The naive solution to this would be to have parties re-randomize everything they receive and add their own noise-message. But this would impose an exponential, in the graph diameter, factor both in the message and communication complexity. Our solution, instead, is to use the homomorphic properties of the encryption scheme and build an efficient process which allows every party to compute an encryption of the OR of the messages it receives from its neighbors. Thus, to transfer a bit $b$, Alice encrypts $b$ and starts flooding, whereas every party encrypts a zero-bit and starts flooding simultaneously. In each following round of the flooding scheme, every party homomorphically computes the OR of the messages it receives and continues flooding with only this encryption. Bob keeps computing the OR of the encryptions he receives, and once sufficiently many rounds have passed, the decryption is invoked to have him obtain Alice's bit. Note that we only treat the case of passively-corrupted parties here, thus no party will input an encryption of a one-bit into this smart flooding scheme which would destroy its correctness.

To solve the second issue—i.e., implement the decryption oracle in a topology-hidingly manner—we introduce a new variant of threshold homomorphic public-key encryption (TH-PKE) with enhanced functionality, which we call *multi-homomorphic threshold encryption with reversible randomization.* Roughly speaking, our new TH-PKE assumes a strongly correlated setup, in which secret (sub)keys are nested in a way which is consistent with the network topology and which allows parties to decrypt messages in a topology-hiding manner. We provide a security definition for the new primitive and describe a topology-hiding protocol for establishing the necessary setup using no setup-assumption whatsoever. And we also describe how to instantiate our schemes under the DDH assumptions. We believe that both the general definition of this augmented TH-PKE and the concrete instantiation could be of independent interest and can be used for anonymizing communication.

**Applications** Building on our topology-hiding network and utilizing the functionality of our topology-hiding homomorphic OR protocol we present the following applications:

**Anonymous broadcast:** We consider a variant of anonymous broadcast where parties can broadcast messages under a pseudonym. The presented protocol allows to realize anonymous broadcast directly from the topology-hiding homomorphic OR protocol without using expensive MPC to setup the pseudonyms.

**Topology-hiding MPC:** Having a topology-hiding network, we can execute on top of it any MPC protocol from the literature that is designed for point-to-point channels which will render it topology-hiding.

### 3.1.3 Comparison with [MOR15]

The work by Moran *et al.* [MOR15] provides the first protocol that solves this problem in the computational setting. Our goals are closely related to theirs. In fact, our security definition of topology-hiding communication and, more general, computation is a refinement of their simulation-based definition of topology-hiding MPC. But our techniques are very different. In light of this similarity in goals, in the following we include a more detailed comparison to our work.

More concretely, the solution of [MOR15] also follows the approach of enhancing the naïve flooding protocol to make it topology-hiding. The key idea is to use nested MPCs, recursively, to protect sensitive information during the execution of the flooding protocol. Roughly, in the basic topology-hiding communication protocol of [MOR15], each party $P_i$ is replaced by a virtual-party $\hat{P}_i$, which is emulated by its immediate neighbors by invoking locally (i.e., in the neighborhood) an off-the-shelf MPC protocol. The complete network of point-to-point channels required by the MPC protocol is emulated by use of a PKE-scheme over the star network centered around $P_i$, i.e., by naïve flooding where $P_i$ is used as the routing node. The above ensures that $P_i$ cannot analyze the messages that are routed through him, as they are actually handled by its corresponding virtual party $\hat{P}_i$. However, there is now a new problem to be solved, namely, how do virtual parties use the underlying (incomplete) communication network to flood messages in a topology-hidingly manner? This is solved as follows: To enable secure communication between adjacent virtual-parties a PKE-scheme is used (once more). Here each virtual-party generates a key-pair and sends the encryption key to the adjacent virtual-parties using real parties as intermediates. This basic protocol is topology-hidingly secure as long as the adversary does not corrupt an entire neighborhood. But this is of course not enough for arbitrarily many corruptions to be tolerated. Thus, to ensure that the overall flooding protocol is also topology-hiding, each virtual party is replaced, again by means of MPC, by a "doubly virtual" party $\hat{\hat{P}}$. This will ensure that only adversaries corrupting all the parties that emulate $\hat{\hat{P}}$ can break the topology-hiding property. To extend the set of tolerable adversaries, the doubly virtual parties are again emulated, and this process is continued until we reach an emulated party that is emulated by all parties in the network. This requires a number of nested MPCs in the order of the network diameter.

In the following we provide a comparison of the solution of [MOR15] with ours, demonstrating the advantages of our solution both in terms of simplicity and efficiency. In all fairness, we should remark that the solution of [MOR15] was explicitly proposed as a proof-of-concept solution. The major advantage of our work over [MOR15] is that our communication protocol makes no use of generic MPC, and makes black-box use of the underlying PKE. This not only yields a substantial efficiency improvement, in terms of both communication and computation, but it also yields a more intuitive solution to the problem, as it uses the natural primitive to make communication private, namely encryption, instead of MPC.

More concretely, the player-virtualization protocol from [MOR15] makes non-black-box use of public-key encryption, i.e., the circuit which is computed via MPC is a public-key encryption/decryption circuit. This is typically a huge circuit which imposes an unrealistic slowdown both on the computation complexity and on the round and/or communication complexity.[3] And this is just at the first level of recursion; the computation of the second level, computes a circuit, which computes the circuit, which computes PK encryptions/decryptions, and so on. Due to the lack of concrete suggestions of instantiation of the PKE and MPC used in [MOR15] we were unable to compute exact estimates on the running time and communication complexity of the suggested protocols. Notwithstanding it should be clear that even for the simple case in which the network has constant degree and logarithmic diameter—for which the communication protocol in [MOR15] achieves a polynomial complexity—and even for the best MPC instantiation the actual constants are huge.

Instead, our solution makes black-box use of the underlying PKE scheme and is, therefore, not only more communication and computation efficient, but also easier to analyze. In fact, in our results we include concrete upper bounds on the communication complexity of all our protocols.

[3]Of course the latter can be traded off by choosing to use either a communication heavy or a round heavy protocol.

Indicatively, for a network with diameter $D$ and maximum degree $d$ our network-hiding broadcast protocol communicates at most $(d+1)^D \cdot n \cdot \lambda$ bits within just $5 \cdot D$ rounds, where $\lambda$ is linear (with small constant, less than 5)[4] in the security parameter $\kappa$ of the underlying PKE scheme. We note that many natural network graphs, such as social networks or the internet have a small diameter.[5]

### 3.1.4 Preliminaries and Notation

We consider an MPC-like setting where $n$ parties $\mathcal{P} = \{P_1, \dots, P_n\}$ wish to communicate in a synchronous manner over some incomplete network of secure channels. When the communication is intended to be from $P_i$, the *sender*, to $P_j$, the *receiver*, we will refer to the parties in $\mathcal{P} \setminus \{P_i, P_j\}$ as the *intermediate parties.* We will assume a passive and static (aka non-adaptive) computationally-bounded adversary who corrupts an arbitrary subset $\overline{H} \subseteq \mathcal{P}$ of parties. We use simulation based security to prove our results. For simplicity our proofs are in Canetti's modular composition framework [Can98] but all our results translate immediately to the universal composition UC framework [Can01] (recall that we consider passive (semi-honest) static security). In fact, to make this transition smoother, we describe our hybrids in the form of UC functionalities. For compactness, for any functionalities $\mathcal{F}$ and $\mathcal{G}$, we will denote by $\{\mathcal{F}, \mathcal{G}\}$ the composite functionality that gives parallel access to $\mathcal{F}$ and $\mathcal{G}$.

### 3.1.5 Outline

The remainder of the chapter is organized as follows. In Section 3.2 we give our definition of topology-hiding security. In Section 3.3 we introduce multi-homomorphic threshold encryption with reversible randomization (RR-MHT-PKE). In Section 3.4 we present a construction based on RR-MHT-PKE which allows to realize topology-hiding communication. First, in Section 3.4.1 we describe a topology-hiding threshold encryption protocol based on RR-MHT-PKE. This protocol is used in Section 3.4.2

[4] This can be contrasted with the complexity $\text{poly}(d)^D \cdot n \cdot \lambda$ obtained by [MOR15].

[5] Backstrom *et al.* [UKBM11] showed that a sub-graph of the Facebook social network consisting of 99.6% of all users had a diameter of 6. In this particular case the broadcast protocol would communicate at most $n^7 \cdot \lambda$ bits within 30 rounds.

to topology-hidingly realize the Boolean-OR functionality. This allows to give a topology-hiding construction of broadcast and secure channels in Section 3.4.3. Finally, in Section 3.5 we present topology-hiding MPC and topology-hiding anonymous broadcast as applications of the protocols from the previous section.

## 3.2 Topology-Hiding Security Definition

In this section we provide the formal simulation-based definition of topology-hiding computation. Our definition is an adaptation of the original simulation-based definition of Moran et al. [MOR15]. More concretely, the topology-hiding property requires that parties learn no information on the underlying communication network other than the description of their local neighborhood, i.e., the identities of their neighbors. To capture this property, we assume that the parties (in the real world) have access to a *network* functionality $\mathcal{N}$ which has knowledge of every party $P_i$'s neighborhood (i.e., the set of point-to-point channels connected to $P_i$) and allows $P_i$ to communicate (only) to its neighbors.

Clearly, a protocol execution over such a network $\mathcal{N}$ allows an adversary using it knowledge of the neighborhood of corrupted parties; thus the simulator needs to also be able to provide this information to its environment. To give this power to the simulator, [MOR15] augments the ideal functionality with an extra component which allows the simulator access to this information. In this work we use $\mathcal{N}$ itself in the ideal world to provide this information to the simulator. Note that this does not affect the security statements, as the trivial $\mathcal{N}$-dummy protocol $\phi^{\mathcal{N}}$ securely realizes $\mathcal{N}$. [6]

A conceptual point in which our model of topology-hiding computation deviates from the formulation of Moran *et al.* has to do with respect to how the communication graph is chosen. At first thought, one might think that parameterizing the network functionality with the communication graph does the trick. This is, however, not the case because the parameters of hybrid-functionalities are known to the protocol which invokes them and are therefore also known to the adversary. The only information

[6]In any case, our protocol will not output anything other than the output of the functionality, hence the simulator will only use $\mathcal{N}$ to learn the corrupted parties neighborhood.

which is not known to the adversary are inputs of honest parties and internal randomness of the functionality; thus, as a second attempt, one might try to have the network functionality sample the communication graph from a given distribution.[7] Unfortunately this also fails to capture the topology-hiding property in full, as we would like to make sure that the adversary (or simulator) gets no information on any *given* (hidden) graph.

Motivated by the above, [MOR15] defines topology-hiding computation using the following trick: they assume an extra incorruptible party, whose only role is to provide the network graph as input to the network functionality. Because this network-choosing party is (by assumption) honest, the simulator cannot see its input and needs to work having only the knowledge that $\mathcal{N}$ allows him to obtain, i.e., the neighborhood of corrupted parties.

In this work we take a slightly different, but equivalent in its effect, approach to avoid the above hack of including a special purpose honest party. We assume that each party provides its desired neighborhood to $\mathcal{N}$ as (a special part of) its input. Since the inputs are explicitly chosen by the environment, we are effectively achieving the same topology-hiding property as [MOR15] but without the extra special-purpose honest party.

In the remainder of this section we provide a formal specification of our network functionality (also referred to as network resource) and our formal security definition of topology-hiding computation.

**The Network** The network topology is captured by means of an undirected graph $G = (V, E)$ with vertex-set $V = \mathcal{P}$ and edge-set $E \subseteq \mathcal{P} \times \mathcal{P}$. An edge $(P_i, P_j) \in E$ indicates that $P_j$ is in the neighborhood of $P_i$, which, intuitively, means that $P_i$ and $P_j$ can communicate over a bilateral secure channel. For a party $P_i$ denote by $\mathbf{N}_G(i)$ its neighborhood in $G$. We will refer to $\mathbf{N}_G[i] = \{P_i\} \cup \mathbf{N}_G(i)$ as $P_i$'s closed neighborhood. Furthermore let $\mathbf{N}_G[i]^k$ be all nodes in $G$ which have distance $k$ or less to $P_i$ (Clearly $P_i \in \mathbf{N}_G[i]^k$.). This set of nodes $\mathbf{N}_G[i]^k$ is called the $k$-neighborhood of $P_i$.

The network functionality allows two types of access: (1) any party $P_i \in \mathcal{P}$ can submit its neighbors $\mathbf{N}_G[i]$, and (2) every party can submit a vector $\vec{m}$ of messages, one for each of its neighbors, which are then

[7]Intuitively, this would correspond to the hidden graph model of [CCG$^+$15].

delivered in a batch form to their intended recipients. In order to be able to make statements for restricted classes of graphs, e.g., expanders, we parameterize the network functionality by a family $\mathcal{G}$ of setups and require that $\mathcal{N}_{\mathcal{G}}$ only allows (the environment on behalf of) the honest parties to chose their neighborhood from this class.

Note, that the adversary is not bound to choose a neighborhood from a graph in $\mathcal{G}$, i.e., any valid neighborhood is accepted for corrupted parties. This is not an issue in the passive setting considered in this work as a passive adversary will submit whatever input the environment hands it. Thus, for the passive case it suffices that the functionality becomes unavailable (halts) upon receiving an invalid neighborhood from the adversary (or from some honest party). [8]

In the description of $\mathcal{N}_{\mathcal{G}}$ we use the following notation: For a graph $G$ with vertex set $V$, and for any $V' \subseteq V$, we denote by $G|_{V'}$ the restriction of $G$ to the vertices in $V'$, i.e., the graph that results by removing from $G$ all vertices in $V \setminus V'$ and their associated edges.

**Functionality $\mathcal{N}_{\mathcal{G}}$**

The network initializes a topology graph $G = (V, E) := (\mathcal{P}, \emptyset)$.

**Info Step:**

1. Every party $P_i \in \mathcal{P}$ (and the adversary on behalf of corrupted parties) sends (input) $(\textsc{MyNeigborhood}, \mathbf{N}_G[i])$ to $\mathcal{N}$; if $\mathbf{N}_G[i]$ is a valid neighborhood for $P_i$, i.e., $\mathbf{N}_G[i] \subseteq \{(P_i, P_j) \mid P_j \in \mathcal{P}\}$, then $\mathcal{N}_{\mathcal{G}}$ updates $E := E \cup \mathbf{N}_G[i]$.
2. If there exist no $G' \in \mathcal{G}$ such that $G' = G$ then $\mathcal{N}_{\mathcal{G}}$ sets $E := \emptyset$ and halts. (Every future input is answered by outputting a special symbol $(\textsc{BadNetwork})$ to the sender of this input.)

**Communication Step:**

1. For each $P_i \in \mathcal{P}$ let $\mathbf{N}_G(i) = \{P_{i_1}, \dots, P_{i_{\nu_i}}\}$.
2. Every $P_i \in \mathcal{P}$ sends $\mathcal{N}_{\mathcal{G}}$ input $(\textsc{send}, \vec{m}_i)$, where $\vec{m}_i = (m_{i,i_1}, \dots, m_{i,i_{\nu_i}})$; if $P_i$ does not submit a vector $\vec{m}_i$ of the right size or format, then $\mathcal{N}_{\mathcal{G}}$ adopts $\vec{m}_i = (\perp, \dots, \perp)$.

[8]Note that the environment knows/chooses all the inputs and therefore knows whether or not the submitted neighborhoods are allowed by the graph class.

3. Every $P_i$ receives (output) $\vec{m}^i = (m_{i_1,i}, \ldots, m_{i_{\nu_i},i})$ from $\mathcal{N}_\mathcal{G}$.

An important feature of the above functionality is that the communication pattern (i.e., which parties send or receive messages) does not reveal to the adversary any information other than the neighborhood of corrupted parties. Thus, the simulator cannot use this functionality in the ideal world to extract information about the network. However, when using this network-functionality (in the real-world protocol) to emulate, e.g., a complete communication network, the adversary might use the messages exchanged in the protocol to extract information that the simulator cannot. In fact, the challenge of a topology-hiding protocol is exactly to ensure that the exchanged messages cannot be used by the adversary in such a way.

**Definition 1.** *Let $\mathcal{G}$ be a family of graphs with vertex set $\mathcal{P}$. Let also $\mathcal{F}$ be a functionality and $\mathcal{N}_\mathcal{G}$ denote the network functionality (as specified above) and $\pi$ be a $\mathcal{N}_\mathcal{G}$-hybrid protocol. We say that $\pi^{\mathcal{N}_\mathcal{G}}$ securely realizes the functionality $\mathcal{F}$ in a topology-hiding manner with respect to network class $\mathcal{G}$ if and only if $\pi$ securely realizes the composite functionality $\{\mathcal{F}, \mathcal{N}_\mathcal{G}\}$.*

## 3.3 Multi-Homomorphic Threshold Encryption with Reversible Randomization

In this section we introduce *multi-homomorphic threshold encryption with reversible randomization* RR-MHT-PKE, a special type of threshold public-key encryption, which will allow us to securely and topology-hidingly realize a distributed encryption scheme. In addition to the (common) homomorphic property of ciphertexts RR-MHT-PKE features homomorphic public-keys and decryption-shares. This allows for a decentralized generation of shared keys which enables parties to generate securely and topology-hidingly a public-key setup where the private-key is shared among all parties. Its reversible randomization property allows parties to transmit public-keys and/or ciphertexts through the network such that the adversary can not track them. We also give a practical implementation of RR-MHT-PKE based on the DDH assumption (see Section 3.3.3).

We first start by recalling some standard definitions. A *public-key encryption* (PKE) scheme consists of three algorithms, $\mathsf{KeyGen}$ for key generation, $\mathsf{Enc}$ for encryption and $\mathsf{Dec}$ for decryption. Since in this work we consider passive adversaries, we will only need encryption satisfying the standard IND-CPA security definition. For completeness this definition is provided in Section 2.2.2.

*Threshold public-key encryption* (T-PKE) with $(\ell, \ell)$-threshold is PKE in which the private key $\mathtt{SK}$ is distributed among $\ell$ parties $P_1, \dots, P_\ell$, such that each party $P_i$ holds a share (aka sub-key) $\mathtt{sk}_i$ of $\mathtt{SK}$ with the property that any $\ell - 1$ sub-keys have no information on $\mathtt{SK}$. Importantly, such a scheme allows for distributed decryption of any given ciphertext: any party $P_i$ can locally compute, using its own sub-key $\mathtt{sk}_i$ of the private key $\mathtt{SK}$, a decryption share $\mathtt{x}_i$, so that if someone gets a hold of decryption shares (for the same $c$) from all parties (i.e., with each of the shares of the private key) he can combine them and recover the plaintext. For the classical definition of T-PKE we refer to Section 2.2.3.

Finally, a *homomorphic threshold public-key encryption* (HT-PKE) scheme is a T-PKE which allows to add up encrypted messages. Here, the message space $\langle \mathcal{M}, + \rangle$ and the ciphertext space $\langle \mathcal{C}, \cdot \rangle$ are groups such that $m_1 + m_2 = \mathsf{Dec}(\mathtt{SK}, \mathsf{Enc}(\mathtt{PK}, m_1; r_1) \cdot \mathsf{Enc}(\mathtt{PK}, m_2; r_2))$. for any key pair $(\mathtt{PK}, \mathtt{SK}) \leftarrow \mathsf{KeyGen}$ and any messages $m_1, m_2 \in \mathcal{M}$.

## 3.3.1 Multi-Homomorphic Threshold Encryption

We first present *multi-homomorphic threshold encryption* which is in essence HT-PKE with two additional properties.

The first property is a decentralized key-generation. The idea is that parties locally generate public/private-key pairs. By combining those local public keys they can then generate a public key with shared private-key where the local private keys act as key shares. More formally, its required that the public-key space $\langle \mathcal{PK}, \cdot \rangle$ and the private-key space $\langle \mathcal{SK}, + \rangle$ are groups. Moreover its is required (1) that there exists a *key-generation algorithm* $\mathsf{KeyGen}$, which outputs a public/private-key pair $(\mathtt{pk}_i, \mathtt{sk}_i) \in \mathcal{PK} \times \mathcal{SK}$, and (2) that for any key pairs $(\mathtt{pk}_1, \mathtt{sk}_1), (\mathtt{pk}_2, \mathtt{sk}_2) \in \mathcal{PK} \times \mathcal{SK}$ it holds that $\mathtt{pk}_1 \cdot \mathtt{pk}_2$ is the public key corresponding to private key $\mathtt{sk}_1 + \mathtt{sk}_2$. In other words a multi-homomorphic threshold encryption scheme is homomorphic with respect to public/private keys. We point out this is not a standard property of threshold PKE schemes. For instance, the threshold variant [DJ01] of Paillier's encryption scheme [Pai99] does not satisfy this property.

Secondly, a multi-homomorphic threshold encryption scheme is required to be homomorphic with respect to decryption shares and private keys. That is, for any key pairs $(\mathtt{pk}_1, \mathtt{sk}_1), (\mathtt{pk}_2, \mathtt{sk}_2)$ and any ciphertext $c$ it must hold that $\mathsf{ShareDecrypt}(\mathtt{sk}_1, c) \cdot \mathsf{ShareDecrypt}(\mathtt{sk}_2, c) = \mathsf{ShareDecrypt}(\mathtt{sk}_1 + \mathtt{sk}_2, c)$.

**Definition 2.** *A* multi-homomorphic threshold encryption *(MHT-PKE) scheme with security parameter $\kappa$ for spaces $\mathcal{M}$, $\mathcal{C}$, $\mathcal{SK}$, and $\mathcal{PK}$ consists of four algorithms $\mathsf{KeyGen}$, $\mathsf{Enc}$, $\mathsf{ShareDecrypt}$, and $\mathsf{Decode}$ which are parametrized by $\kappa$ where:*

1. *The (probabilistic)* key-generation *algorithm $\mathsf{KeyGen}$ outputs a public key $\mathtt{pk} \in \mathcal{PK}$ and a private key $\mathtt{sk} \in \mathcal{SK}$.*

2. Homomorphic keys: *The public-key space $\langle \mathcal{PK}; \cdot \rangle$ and the private-key space $\langle \mathcal{SK}; + \rangle$ are cyclic groups of prime order. For any key pairs $(\mathtt{pk}_1, \mathtt{sk}_1)$, $(\mathtt{pk}_2, \mathtt{sk}_2) \in \mathcal{PK} \times \mathcal{SK}$ it holds that $\mathtt{pk}_1 \cdot \mathtt{pk}_2$ is the public key corresponding to private key $\mathtt{sk}_1 + \mathtt{sk}_2$.*

3. *The (probabilistic)* encryption algorithm *$\mathsf{Enc}$ takes a public key $\mathtt{pk} \in \mathcal{PK}$ and a message $m \in \mathcal{M}$ and outputs a ciphertext $c \leftarrow \mathsf{Enc}(\mathtt{pk}, m; r)$.*

4. Homomorphic ciphertexts: *The message space $\langle \mathcal{M}; + \rangle$ and the ciphertext space $\langle \mathcal{C}; \cdot \rangle$ are cyclic groups of prime order. For any public-key $\mathtt{pk} \in \mathcal{PK}$ and any two messages $m_1, m_2 \in \mathcal{M}$ where $c_1 \leftarrow \mathsf{Enc}(\mathtt{pk}, m_1; r_1)$ and $c_2 \leftarrow \mathsf{Enc}(\mathtt{pk}, m_2; r_2)$ it holds that $c_1 \cdot c_2$ is an encryption of $m_1 + m_2$ under $\mathtt{pk}$.*

5. *The* decryption share algorithm *$\mathsf{ShareDecrypt}$ takes a private key $\mathtt{sk}_i \in \mathcal{SK}$ and a ciphertext $c \in \mathcal{C}$ as inputs and outputs a decryption share $\mathtt{x}_i \leftarrow \mathsf{ShareDecrypt}(\mathtt{sk}_i, c)$.*

6. Homomorphic decryption-shares: *The decryption-share space $\langle \mathcal{DS}; \cdot \rangle$ is a cyclic group of prime order. For ciphertext $c \in \mathcal{C}$ and private keys $\mathtt{sk}_1, \mathtt{sk}_2 \in \mathcal{SK}$ where $\mathtt{x}_1 \leftarrow \mathsf{ShareDecrypt}(\mathtt{sk}_1, c)$ and $\mathtt{x}_2 \leftarrow \mathsf{ShareDecrypt}(\mathtt{sk}_2, c)$ we have $\mathtt{x}_1 \cdot \mathtt{x}_2 = \mathsf{ShareDecrypt}(\mathtt{sk}_1 + \mathtt{sk}_2, c)$.*

7. *The* decoding algorithm $\mathsf{Decode}$ *takes a decryption share* $\boldsymbol{x} \in \mathcal{DS}$ *and a ciphertext* $c \in \mathcal{C}$ *and outputs a message* $m \leftarrow \mathsf{Decode}(\boldsymbol{x}, c)$*.*

*A MHT-PKE scheme is* correct *if it satisfies the following correctness property: For any key pairs* $(\boldsymbol{pk}_1, \boldsymbol{sk}_1), \ldots, (\boldsymbol{pk}_l, \boldsymbol{sk}_l) \leftarrow \mathsf{KeyGen}$ *and any message* $m \in \mathcal{M}$ *it holds that* $m = \mathsf{Decode}(\boldsymbol{x}_1 \cdot \ldots \cdot \boldsymbol{x}_l, c)$ *where* $\boldsymbol{x}_i = \mathsf{ShareDecrypt}(\boldsymbol{sk}_i, c)$*,* $c = \mathsf{Enc}(\boldsymbol{pk}, m; r)$ *and* $\boldsymbol{pk} = \boldsymbol{pk}_1 \cdot \ldots \cdot \boldsymbol{pk}_l$*. We assume, that the order of each of the above groups is publicly-known and that the group operations are efficiently computable. Moreover, we require that given a message* $m \in \mathcal{M}$ *and a ciphertext* $c \in \mathcal{C}$ *one can efficiently invert* $\mathsf{Decode}$*, i.e., compute a decryption share* $\boldsymbol{x}$ *with* $m = \mathsf{Decode}(\boldsymbol{x}, c)$*.*

We define the security of MHT-PKE with respect to a threshold variant of the IND-CPA security definition.

**Definition 3.** *A MHT-PKE scheme is* IND-TCPA *secure if the adversary's advantage in winning the following game is negligible in* $\kappa$*.*

1. *The game generates key pairs* $(\boldsymbol{pk}_1, \boldsymbol{sk}_1), \ldots (\boldsymbol{pk}_l, \boldsymbol{sk}_l) \twoheadleftarrow \mathsf{KeyGen}$ *and chooses a random bit* $b$*. Then the adversary gets* $\boldsymbol{pk} = \boldsymbol{pk}_1 \cdot \ldots \cdot \boldsymbol{pk}_l$*,* $\boldsymbol{pk}_1, \ldots, \boldsymbol{pk}_l$ *and* $\boldsymbol{sk}_2, \ldots, \boldsymbol{sk}_l$*. This allows him to generate encryptions of arbitrary messages and to generate decryption shares for all key pairs except* $(\boldsymbol{pk}_1, \boldsymbol{sk}_1)$*.*

2. *The adversary specifies two messages* $m_0$ *and* $m_1$ *and the game returns* $c = \mathsf{Enc}(\mathit{PK}, m_b)$*.*

3. *The adversary specifies a bit* $b'$*. If* $b = b'$ *the adversary has won the game.*

Furthermore for any chosen public-key $\mathtt{pk} \in \mathcal{PK}$, it should be hard to distinguish between $(\mathtt{pk}, \mathtt{pk} \cdot \mathtt{pk}_1)$ and $(\mathtt{pk}, \mathtt{pk}_2)$ where $\mathtt{pk}_1, \mathtt{pk}_2$ are distributed according to $\mathsf{KeyGen}$. This is the case, if the output distribution of $\mathsf{KeyGen}$ is indistinguishable from uniform.

**Definition 4.** *A MHT-PKE scheme has the* unif-KG *property if the output distribution of* $\mathsf{KeyGen}$ *is indistinguishable from the uniform distribution.*

### 3.3.2 Reversible Randomization

In this section we introduce multi-homomorphic threshold encryption with *reversible randomization* which is MHT-PKE with additional randomization properties for public-keys and ciphertexts.

**Randomization of Public-Keys.** The first property required is the randomization of public-keys. More concretely, a MHT-PKE with reversible randomization allows a party $P_i$ with public key $\texttt{pk}_i$ to "randomize" $\texttt{pk}_i$, i.e., compute a new masked public-key $\widetilde{\texttt{pk}}_i$ so that anyone seeing $\widetilde{\texttt{pk}}_i$ is unable to tell whether it is a freshly generated public-key or a randomized version of $\texttt{pk}_i$. Importantly, we require the randomization algorithm to be reversible in the following sense. The randomization algorithm must provide $P_i$ with information $\texttt{rk}_i$, the *de-randomizer*, which allows it to map any encryption with $\widetilde{\texttt{pk}}_i$ back to an encryption with its original key $\texttt{pk}_i$. Looking ahead, the randomization of public-keys property will ensure that the adversary can not trace public keys while they travel the network. This allows us to build a topology-hiding information-transmission protocol.

**Randomization of Ciphertexts.** The second property required is the randomization of ciphertexts. More concretely, a MHT-PKE with reversible randomization allows a party $P_i$ with ciphertext $c_i$ to "randomize" $c_i$, i.e., compute a new masked ciphertext $\widehat{c}_i$ so that anyone seeing $\widehat{c}_i$ is unable to tell whether it is a freshly generated ciphertext (using an arbitrary public-key) or an randomized version of $c_i$. Importantly, we require the randomization algorithm to be reversible. This means it must provide $P_i$ with information $\texttt{rk}_i$, the *de-randomizer*, which allows it to map any decryption share of $\widehat{c}_i$ and decryption key $\texttt{sk}$ back to a decryption share of the original ciphertext $c_i$ and $\texttt{sk}$. Looking ahead, the randomization of ciphertexts will ensure that the adversary can not trace ciphertexts and decryption-shares while they travel the network. This will allow us to build a topology-hiding decryption protocol. We remark that this property differs from the usual ciphertext re-randomization in homomorphic PKE schemes where one randomizes a ciphertext by adding up an encryption of 0.

**MHT-PKE with Reversible Randomization** We can now give the formal definition of *multi-homomorphic threshold encryption with reversible-randomization* (RR-MHT-PKE).

**Definition 5.** *A RR-MHT-PKE scheme is a MHT-PKE scheme with four extra algorithms* $\mathsf{RandKey}$, $\mathsf{DerandCipher}$, $\mathsf{RandCipher}$, $\mathsf{DerandShare}$ *where:*

1. *The (probabilistic)* (key) randomization algorithm $\mathsf{RandKey}$ *takes a public key* $\boldsymbol{pk} \in \mathcal{PK}$ *and outputs a new public key* $\widetilde{\boldsymbol{pk}} \in \mathcal{PK}$ *and* a de-randomizer $\boldsymbol{rk} \in \mathcal{RK}_P$.

2. *The* (ciphertext) de-randomization algorithm $\mathsf{DerandCipher}$ *takes a de-randomizer* $\boldsymbol{rk} \in \mathcal{RK}_P$ *and a ciphertext* $\widetilde{c} \in \mathcal{C}$ *and outputs a new ciphertext* $c \in \mathcal{C}$ *such that the following property holds. For any key pair* $(\boldsymbol{pk}, \boldsymbol{sk})$, $(\widetilde{\boldsymbol{pk}}, \boldsymbol{rk}) \leftarrow \mathsf{RandKey}(\boldsymbol{pk}; r')$, *any message* $m \in \mathcal{M}$, *and any ciphertext* $\widetilde{c} \leftarrow \mathsf{Enc}(\widetilde{\boldsymbol{pk}}, m; \tilde{r})$ *there exists an* $r$ *such that* $\mathsf{Enc}(\boldsymbol{pk}, m; r) = \mathsf{DerandCipher}(\boldsymbol{rk}, \widetilde{c})$. *Moreover, given a ciphertext* $c$ *and a de-randomizer* $\boldsymbol{rk}$ *one can efficiently invert* $\mathsf{DerandCipher}$, *i.e., compute a ciphertext* $\widetilde{c}$ *such that* $c = \mathsf{DerandCipher}(\boldsymbol{rk}, \widetilde{c})$.

3. *The (probabilistic)* (ciphertext) randomization algorithm $\mathsf{RandCipher}$ *takes a ciphertext* $c \in \mathcal{C}$ *and outputs a new ciphertext* $\hat{c} \in \mathcal{C}$ *and a de-randomizer* $\boldsymbol{rk} \in \mathcal{RK}_C$.

4. *The* (share) de-randomization algorithm $\mathsf{DerandShare}$ *takes a de-randomizer* $\boldsymbol{rk} \in \mathcal{RK}_C$ *and a decryption share* $\hat{\boldsymbol{x}} \in \mathcal{DS}$ *and outputs a share* $\boldsymbol{x} \in \mathcal{DS}$ *such that the following property holds. For any key pair* $(\boldsymbol{pk}, \boldsymbol{sk})$, *any ciphertext* $c \in \mathcal{C}$, *the pair* $(\boldsymbol{rk}, \hat{c}) \leftarrow \mathsf{RandCipher}(c; r)$, *and decryption share* $\hat{\boldsymbol{x}} \leftarrow \mathsf{ShareDecrypt}(\boldsymbol{sk}_i, \hat{c})$ *we have that* $\mathsf{DerandShare}(\boldsymbol{rk}, \hat{\boldsymbol{x}}) = \mathsf{ShareDecrypt}(\boldsymbol{sk}_i, c)$. *Moreover, given a decryption share* $\boldsymbol{x}$ *and a de-randomizer* $\boldsymbol{rk}$ *one can efficiently invert* $\mathsf{DerandShare}$, *i.e., compute a decryption shares* $\hat{\boldsymbol{x}}$ *such that* $\boldsymbol{x} = \mathsf{DerandShare}(\boldsymbol{rk}, \hat{\boldsymbol{x}})$.

For any public key $\mathtt{pk}$ it should be hard (for the adversary) to distinguish between $(\mathtt{pk}, \mathsf{RandKey}(\mathtt{pk}))$ and $(\mathtt{pk}, \mathtt{pk}')$ where $\mathtt{pk}'$ is freshly generated using $\mathsf{KeyGen}$. Similar, for any ciphertext $c$ it should be hard to distinguish between $(c, \mathsf{RandCipher}(c))$ and $(c, c')$ where $c'$ is a randomly chosen ciphertext. More formally, the scheme should have the *indistinguishability under chosen public-key and chosen ciphertext attack* (IND-CKCA) property.

**Definition 6.** *A RR-MHT-PKE scheme is* IND-CKCA *secure if the probability of the adversary winning the following game is negligible (in $\kappa$) close to $\frac{3}{4}$.*

1. *The adversary specifies a public key $\mathsf{pk} \in \mathcal{PK}$ and a ciphertext $c \in \mathcal{C}$.*
2. *The game generates key pairs $(\mathsf{pk}_1, \mathsf{sk}_1), (\mathsf{pk}_2, \mathsf{sk}_2) \twoheadleftarrow \mathsf{KeyGen}$ and a uniform random message $m \in \mathcal{M}$. The game then chooses uniform random bits $b_1$ and $b_2$. The adversary gets public key $\widetilde{\mathsf{pk}}$ and ciphertext $\widehat{c}$ where*

   $$\widetilde{\mathsf{pk}} = \begin{cases} \mathsf{RandKey}(\mathsf{pk}) & \text{if } b_1 = 0 \\ \mathsf{pk}_1 & \text{if } b_1 = 1 \end{cases}$$

   *and*

   $$\widehat{c} = \begin{cases} \mathsf{RandCipher}(c) & \text{if } b_2 = 0 \\ \mathsf{Enc}(\mathsf{pk}_2, m) & \text{if } b_2 = 1 \end{cases} .$$

3. *The adversary specifies bits $b'_1$ and $b'_2$. If $b_1 = b'_1$ or $b_2 = b'_2$ the adversary has won the game.*

The security of a RR-MHT-PKE scheme is defined with respect to the above security properties.

**Definition 7.** *A multi-homomorphic threshold encryption with reversible-randomization scheme is* secure *if it is* IND-TCPA, *and* IND-CKCA *secure and has the* unif-KG *property.*

### 3.3.3 RR-MHT-PKE based on DDH

In this section we present a secure RR-MHT-PKE scheme based on the DDH assumption. Our construction can be seen as an extended variant of the ElGamal cryptosystem [ElG84].

**MHT-PKE-Algorithms.**

Let $\langle G, \cdot \rangle$ be a cyclic group of prime order $q(\kappa)$ (where the bit length of $q$ is $\text{poly}(\kappa)$) for security parameter $\kappa$. Denote by $e_G$ the neutral element of $G$ and let $g$ be a generator of $G$. For our scheme we assume that $(G, q, g)$ is publicly known. To ensure the various homomorphic properties we use $G$ as message space, public key space, decryption share space, i.e., $\mathcal{M} = \mathcal{PK} = \mathcal{DS} = G$. Ciphertexts consist of two group elements, i.e., $\mathcal{C} = G \times G$. Moreover, the private-key space is $\mathcal{SK} = \langle \{0, \dots, p-1\}, + \rangle$. The core part of our construction are the following four MHT-PKE algorithms.

**Key Generation:** Key-generation algorithm $\mathsf{KeyGen}$ chooses a private-key $\mathtt{sk}$ uniformly at random from $\mathcal{SK}$ and sets the public-key $\mathtt{pk} = g^{\mathtt{sk}}$, i.e.,

$$(g^{\mathtt{sk}}, \mathtt{sk}) \twoheadleftarrow \mathsf{KeyGen}().$$

**Encryption:** The encryption algorithm $\mathsf{Enc}$ encrypts message $m$ under public-key $\mathtt{pk}$ as $c := (g^r, \mathtt{pk}^r \cdot m)$ where $r \in \{1, \dots, p-1\}$ is chosen uniformly at random.

$$\mathsf{Enc}(\mathtt{pk}, m; r) = (g^r, \mathtt{pk}^r \cdot m)$$

**Decryption Shares:** Decryption share algorithm $\mathsf{ShareDecrypt}$ takes a ciphertext $c = (a, b)$ and a private-key $\mathtt{sk}$ and computes decryption share $\mathtt{x} = a^{-\mathtt{sk}}$, i.e.,

$$\mathsf{ShareDecrypt}(\mathtt{sk}, c = (a, b)) = a^{-\mathtt{sk}}.$$

**Decode:** The decoding algorithm $\mathsf{Decode}$ takes a ciphertext $c = (a, b)$ and a decryption share $\mathtt{x}$ and computes message $m = \mathtt{x} \cdot b$, i.e.,

$$\mathsf{Decode}(\mathtt{x}, c = (a, b)) = \mathtt{x} \cdot b.$$

Note also that $\mathtt{x} = m \cdot b^{-1}$, i.e., $\mathsf{Decode}$ is efficiently invertible.

It is easy to show that those four algorithms satisfy the correctness property required by Definition 2. In the following we assume that decisional Diffie-Hellman (DDH) assumption holds in $G$ (cf. Section 2.3.1).

A simple choice for $G$ is a Schnorr Group, which is a $q$-order subgroup of $\mathbb{Z}_p^\times$ where $p, q$ are primes with $p = qr + 1$ for some $r$. A more efficient and therefore preferred alternative is to use an appropriate elliptic curve group (see, e.g., [Bon98]).

**Lemma 1.** *The above MHT-PKE scheme is IND-TCPA secure if DDH holds for $G$.*

*Proof.* We give a reduction of DDH to IND-TCPA. Consider a winner $\widetilde{W}$ for the IND-TCPA game (cf. Definition 3). Then the following reduction gives a winner $W$ for DDH.

**Reduction DDH to IND-TCPA**

1. On input of $(\alpha, \beta, \gamma)$ from the outside set $\mathtt{pk} := \alpha$, choose a random bit $b$, generate key pairs $(\mathtt{pk}_2, \mathtt{sk}_2), \dots, (\mathtt{pk}_l, \mathtt{sk}_l) \leftarrow \mathsf{KeyGen}$, and set $\mathtt{pk}_1 := \mathtt{pk} \cdot g^{-\mathtt{sk}_2} \cdot \ \dots \ \cdot g^{-\mathtt{sk}_l}$. Then send $\mathtt{pk}$, $\mathtt{pk}_1, \dots, \mathtt{pk}_l$ and $\mathtt{sk}_2, \dots, \mathtt{sk}_l$ to the winner $\widetilde{W}$.
2. Given the messages $m_0$ and $m_1$ from $\widetilde{W}$ return $(\beta, \gamma \cdot m_b)$ to $\widetilde{W}$.
3. If $\widetilde{W}$ guesses $b$ correctly output 1, else output 0.

If the input $(\alpha, \beta, \gamma)$ is of the form $(g^a, g^b, g^{ab})$ the reduction returns in the second step $(g^b, g^{ab} \cdot m_b) = (g^b, \mathtt{pk}^b \cdot m_b)$ to $\widetilde{W}$ which is a valid encryption of $m_b$ under $\mathtt{pk} := \alpha$. The winner $\widetilde{W}$ should therefore be able to distinguish between $(\beta, \gamma \cdot m_0)$ and $(\beta, \gamma \cdot m_1)$ with non-negligible advantage. If $\gamma$ is uniform at random the tuple $(\beta, \gamma \cdot m_b)$ is independent of bit $b$. The winner $\widetilde{W}$ should therefore have a negligible advantage in guessing the bit correctly. □

**Lemma 2.** *The above MHT-PKE scheme has the* unif-KG *property.*

*Proof.* $\mathsf{KeyGen}$ chooses the private-key $\mathtt{sk}$ uniformly at random from $\mathcal{SK}$. Thus the output public-key is a uniform random element from $\mathcal{PK}$. □

**Reversible-Randomization Algorithms.**

The final part of our construction are the four (de)randomization algorithms which complete the RR-MHT-PKE scheme. The used de-randomizer spaces are $\mathcal{RK}_P = \mathcal{RK}_C = \{0, \dots, p-1\}$.

**Public-Key Randomization:** To randomize a public-key $\mathtt{pk}$ the key randomization algorithm $\mathsf{RandKey}$ generates a key pair $(g^{\mathtt{rk}}, \mathtt{rk}) \twoheadleftarrow \mathsf{KeyGen}$ and multiplies $\mathtt{pk}$ with $g^{\mathtt{rk}}$. The private value $\mathtt{rk}$ acts as the de-randomizer.

$$(\widetilde{\mathtt{pk}}, \mathtt{rk}) = (g^{\mathtt{rk}} \cdot \mathtt{pk}, \mathtt{rk}) \twoheadleftarrow \mathsf{RandKey}(\mathtt{pk})$$

**Ciphertext De-Randomization:** To de-randomize a given ciphertext $\widetilde{c}$ the de-randomization algorithm $\mathsf{DerandCipher}$ essentially computes a decryption share for $\mathtt{rk}$ and strips it from $\widetilde{c}$.

$$\mathsf{DerandCipher}\big(\mathtt{rk}, \widetilde{c} = (a, b)\big) = (a, a^{-\mathtt{rk}} \cdot b)$$

We observe that $\mathsf{DerandCipher}$ is efficiently invertible.

**Ciphertext Randomization:** To randomize a ciphertext $c = (a, b)$ the algorithm $\mathsf{RandCipher}$ chooses a random $r, r' \in G$, and a random $d \in \{1, \dots, p-1\}$ and computes[9] $e \in \{1, \dots, p-1\}$ with $e \cdot d \equiv_p 1$. If $a \neq e_G$ it exponentiates $a$ with $e$, otherwise it replaces $a$ by $r$.

$$(\hat{c}, \mathtt{rk}) = \begin{cases} \big((a^e, r'), d\big) & \text{if } a \neq e_G \\ \big((r\ , r'), 0\big) & \text{if } a = e_G \end{cases} \twoheadleftarrow \mathsf{RandCipher}\big(c = (a, b)\big)$$

**Decryption-Share De-Randomization:** To de-randomize a given decryption share $\hat{\mathtt{x}}$ the algorithm computes $\hat{\mathtt{x}}^{\mathtt{rk}}$.

$$\mathsf{DerandShare}(\mathtt{rk}, \hat{\mathtt{x}}) = \hat{\mathtt{x}}^{\mathtt{rk}}$$

We observe that $\mathsf{DerandShare}$ is efficiently invertible.

We can now show that the above algorithms satisfy the correctness properties of Definition 5. For a public key $\mathtt{pk}$ let $(\widetilde{\mathtt{pk}}, \mathtt{rk}) \twoheadleftarrow \mathsf{RandKey}(\mathtt{pk})$. For any message $m$ consider the encryption $\tilde{c} = \mathsf{Enc}(\widetilde{\mathtt{pk}}, m; r)$ of $m$ under public-key $\widetilde{\mathtt{pk}}$ with randomness $r$. Then we have

$$\begin{aligned} \mathsf{DerandCipher}(\mathtt{rk}, \tilde{c}) &= (g^r, (g^r)^{-\mathtt{rk}} \cdot g^{\mathtt{rk} \cdot r} \cdot \mathtt{pk}^r \cdot m) \\ &= (g^r, \mathtt{pk}^r \cdot m) = \mathsf{Enc}(\mathtt{pk}, m; r). \end{aligned}$$

[9]This can be done efficiently using the Extended Euclidean algorithm.

Next, for a ciphertext $c = (a, b)$ let $\big(\widehat{c} = (\widehat{a}, \widehat{b}), \texttt{rk}\big) \leftarrow \mathsf{RandCipher}(c)$. For an arbitrary private-key $\texttt{sk}$ let $\hat{\texttt{x}} = \mathsf{ShareDecrypt}\big(\texttt{sk}, \widehat{c}\big)$ be the decryption-share for $\widehat{c}$. Then we have

$$\mathsf{DerandShare}(\texttt{rk}, \widehat{\texttt{x}}) = \widehat{\texttt{x}}^{\texttt{rk}} = \widehat{a}^{-\texttt{sk}\cdot\texttt{rk}}$$
$$= \begin{cases} a^{-\texttt{sk}\cdot e\cdot d} = a^{-\texttt{sk}} = \mathsf{ShareDecrypt}(\texttt{sk}, c) & \text{if } a \neq e_G \\ r^{-\texttt{sk}\cdot 0} = e_G = e_G^{-\texttt{sk}} = \mathsf{ShareDecrypt}(\texttt{sk}, c) & \text{if } a = e_G. \end{cases}$$

The correctness properties are thus fulfilled. It remains to show that the scheme satisfies the IND-CKCA security property.

**Lemma 3.** *The above RR-MHT-PKE scheme is IND-CKCA secure.*

*Proof.* For any public key pair $(\widetilde{\texttt{pk}}, \texttt{pk})$ there exists a $\texttt{rk} \in \mathcal{RK}_P$ such that $\widetilde{\texttt{pk}} = g^{\texttt{rk}} \cdot \texttt{pk}$. The key pair $(\widetilde{\texttt{pk}}, \texttt{pk})$ is thus distributed interdependently from $b_1$. The adversary thus has a negligible advantage in guessing the $b_1$ correctly. Consider ciphertext pair $\big(\widehat{c} = (\widehat{a}, \widehat{b}), c = (a, b)\big)$. As long as $a = e_G$ or $\widehat{a} \neq e_G$ there exist $r, r', d$ such that $(\hat{c}, \texttt{rk}) = \mathsf{RandCipher}(c = (a, b); r, r', d)$. In this case the adversary thus has a negligible advantage in guessing the $b_2$ correctly. If $a = e_G$ and $\widehat{a} = e_G$ the ciphertext $\widehat{c}$ cannot be the randomization of $c$ and guessing $b_2$ is easy. However, the probability that $a = e_G$ and $\widehat{a} = e_G$ is negligible for $q(\kappa)$ large enough. □

**Lemma 4.** *The above RR-MHT-PKE scheme is a secure.*

*Proof.* Follows directly from Definition 7 and Lemmas 1,2, and 3. □

## 3.4 Topology-Hiding Communication

In this section we present a construction which allows to securely and topology-hidingly realize different types of communication channels using black-box PKE. The section consists of the following four steps, each treated in a separate sub-section.

**Topology-Hiding Encryption.** In Section 3.4.1, a topology-hiding threshold encryption protocol based on *black-box* RR-MHT-PKE is presented. More precisely, we provide (1) a distributed setup protocol, (2) an information-transmission protocol, and (3) a distributed decryption protocol.

**Topology-Hiding Boolean-OR.** In Section 3.4.2 we present a protocol which, for networks with moderate degree and diameter, securely and topology-hidingly realizes the multi-party Boolean-OR functionality using the topology-hiding threshold encryption protocol from the previous section.

**Topology-Hiding Broadcast and Secure Channels.** Finally, in Section 3.4.3 we use the Boolean-OR functionality to securely and topology-hidingly realize secure channels and broadcast. The main result of this section is the following theorem.

**Theorem 1.** *Given a network $\mathcal{N}_{\mathcal{G}}$ with diameter $D$ and maximum degree $d$ where $d^D = \mathrm{poly}(\kappa)$ there exists a protocol which securely and topology-hidingly realizes broadcast using black-box RR-MHT-PKE. The protocol communicates at most $(d+1)^D \cdot n \cdot \lambda$ bits within $5 \cdot D$ rounds, where $\lambda$ is linear (with small constant, less than $5$) in $\kappa$.*

## 3.4.1 Topology-Hiding Threshold Encryption

In this section we build a topology-hiding threshold encryption protocol using a secure RR-MHT-PKE scheme. More precisely, we provide (1) a distributed setup protocol, (2) an information-transmission protocol, and (3) a distributed decryption protocol. Looking ahead, those protocols will allow us to topology-hidingly realize the Boolean-OR functionality.

**The RR-MHT-PKE Scheme.** We assume that the parties have access to a secure RR-MHT-PKE scheme with security parameter $\kappa$, where $n = \mathrm{poly}(\kappa)$. In particular, each party has local (black-box) access to the algorithms of the RR-MHT-PKE scheme. We refer to Section 3.3.3 for secure a RR-MHT-PKE scheme based on the DDH assumption.

**The Network Graph.** A prerequisite for our protocols to work is that the network graph $G$ of $\mathcal{N}_{\mathcal{G}}$ is connected. Otherwise (global) information transmission is not possible. The parties also need to know upper bounds on the maximum degree and the diameter of the network graph. We therefore assume that the parties have access to an initialized network $\mathcal{N}_{\mathcal{G}}^{d,D}$ where the graphs in the family $\mathcal{G}$ are connected, have a maximum degree of $d \leq n$, and a diameter of at most $D \leq n$ where $d$ and $D$ are publicly known. For simplicity we restrict ourselves to present protocols for $d$-regular network graphs. We point out that one can extend the presented protocols to the general case where parties may have less than $d$ neighbors. The idea is that a party which has less than $d$ neighbors pretends to have $d$ neighbors by emulating (messages from) virtual neighbors (cf. [MOR15]).

## Setup Protocol

In this section we present a protocol which allows to topology-hidingly generate a threshold-setup where each party $P_i$ holds a public key $\texttt{PK}_i$ such that the corresponding private-key is shared among all parties. The high-level idea of our protocol is as follows. We first observe that the $D$-neighborhood of $P_i$ consists of all parties. The setup provides party $P_i$ with a public key where the corresponding private-key is shared among the parties in the $D$-neighborhood $\mathbf{N}_G[i]^D$ of $P_i$. The setup can be generated recursively. In order to generate a $k$-neighborhood public-key $\texttt{PK}_i^{(k)}$, $P_i$ asks each of its neighbors to generate a public key where the private key is shared in the neighbor's $(k-1)$-neighborhood. It can then compute $\texttt{PK}_i^{(k)}$ by combining the received public-keys.

**Definition 8.** *A* setup for topology-hiding threshold encryption over a network $\mathcal{N}_G^{d,D}$ *consists of the following parts.*

**Private-Key Shares:** *Each party $P_i$ holds a vector $(\overline{SK}_i^{(0)}, \ldots, \overline{SK}_i^{(D)})$ of $D+1$ private keys which we call its* private-key shares*. For any $0 \leq k \leq D$ we denote by $\overline{PK}_i^{(k)}$ the public key corresponding to $\overline{SK}_i^{(r)}$.*

**Public-Keys:** *Each party $P_i$ holds $D+1$ public keys $(PK_i^{(0)}, \ldots, PK_i^{(D)})$ where $PK_i^{(0)} = \overline{PK}_i^{(0)}$ and $PK_i^{(k)} = \overline{PK}_i^{(k)} \cdot \prod_{P_j \in \mathbf{N}_G(i)} PK_j^{(k-1)}$. We call $PK_i^{(k)}$ the* level-$k$ public-key *of $P_i$ and denote by $SK_i^{(r)}$ the corresponding (shared) private key. The* public-key *of $P_i$ is $PK_i := PK_i^{(D)}$ and the* shared private-key *is $SK_i := SK_i^{(D)}$.*

**Local Pseudonyms:** *Each party $P_i$ privately holds an injective random function $\nu_i(\cdot) : \mathbf{N}_G(i) \to \{1, \ldots, d\}$ which assigns each neighbor $P_j \in \mathbf{N}_G(i)$ a unique local identity $\nu_i(j) \in \{1, \ldots, d\}$. W.l.o.g. we will assume that $\nu_i(i) = 0$.*

We remark that the condition on the public-keys ensures that any $0 \leq k \leq D$ the private key $\mathtt{SK}_i^{(k)}$ is properly shared among the $k$ neighborhood of $P_i$, i.e., each party in the $k$-neighborhood holds a non-trivial share.

**Definition 9.** *A protocol is a* secure (topology-hiding) setup protocol over a network $\mathcal{N}_G^{d,D}$ *if it has the following properties.*

**Correctness:** *The protocol generates with overwhelming probability a setup for topology-hiding threshold encryption over network $\mathcal{N}_G^{d,D}$.*

**Topology-Hiding Simulation:** *The adversarial view in a protocol execution can be simulated with overwhelming probability given the neighborhood of dishonest parties in $\mathcal{N}_G^{d,D}$ and the output of dishonest parties, i.e., given the values*

$$\{\mathbf{N}_G(i), \nu_i(\cdot), \overline{SK}_i^{(0)}, \dots, \overline{SK}_i^{(D)}, PK_i^{(0)}, \dots, PK_i^{(D)}\}_{P_i \in \overline{H}}$$

The simulation property ensures in particular that (a) the adversary does not learn more about the network topology and that (b) the adversary does not learn the private key corresponding to the public key $\mathtt{PK}_i^{(k)}$ of party $P_i$ unless it corrupts the entire $k$-neighborhood of $P_i$.

**Protocol GenerateSetup**

```
Require: Parties have access to an initialized $\mathcal{N}_G^{d,D}$.
1: Each $P_i$ generates the local identities $\nu_i(\cdot)$ and sub-key pair
   $(\overline{\mathtt{PK}}_i^{(0)}, \overline{\mathtt{SK}}_i^{(0)}) \twoheadleftarrow \mathsf{KeyGen}$. Then it sets $\mathtt{PK}_i^{(0)} = \overline{\mathtt{PK}}_i^{(0)}$.
2: for $k = 1, \dots, D$ do
3:     Each $P_i$ sends $\mathtt{PK}_i^{(k-1)}$ to each $P_j \in \mathbf{N}_G(i)$ using $\mathcal{N}$.
4:     Each $P_i$ generates sub-key pair $(\overline{\mathtt{PK}}_i^{(k)}, \overline{\mathtt{SK}}_i^{(k)}) \twoheadleftarrow \mathsf{KeyGen}$.
5:     Each $P_i$ computes $\mathtt{PK}_i^{(k)} = \overline{\mathtt{PK}}_i^{(k)} \cdot \prod_{P_j \in \mathbf{N}_G(i)} \mathtt{PK}_j^{(k-1)}$.
6: end for
Output: $P_i$ outputs $\nu_i(\cdot)$, $(\overline{\mathtt{SK}}_i^{(0)}, \dots, \overline{\mathtt{SK}}_i^{(D)})$, and $(\mathtt{PK}_i^{(0)}, \dots, \mathtt{PK}_i^{(D)})$.
```

**Lemma 5.** *Given a secure RR-MHT-PKE scheme protocol* GenerateSetup *is a secure setup protocol. The protocol communicates $D \cdot d \cdot n \cdot \log|\mathcal{PK}|$ bits within $D$ rounds.*

*Proof. Correctness:* It follows directly from protocol inspection that the setup generated by GenerateSetup is valid for $\mathcal{N}_{\mathcal{G}}^{d,D}$.

*Topology-Hiding Simulation:* The view of the adversary during an actual protocol execution is

$$\mathtt{view}\left\{\begin{array}{l}\mathbf{N}_G(i), \nu_i(\cdot), \left\{\mathtt{PK}_i^{(k)}, \overline{\mathtt{PK}}_i^{(k)}, \overline{\mathtt{SK}}_i^{(k)}\right\}_{0\le r\le D}, \\ \left\{\mathtt{PK}_j^{(k)}\right\}_{P_j\in\mathbf{N}_G(i), 0\le r\le D-1}\end{array}\right\}_{P_i\in\overline{H}}.$$

Now consider the view where public keys $\left\{\mathtt{PK}_j^{(k)}\right\}_{P_j\in\mathbf{N}_G(i)\cap H, 0\le r\le D-1}$ are replaced by freshly generated public keys using KeyGen, i.e.,

$$\mathtt{view}\left\{\begin{array}{l}\mathbf{N}_G(i), \nu_i(\cdot), \left\{\mathtt{PK}_i^{(k)}, \overline{\mathtt{PK}}_i^{(k)}, \overline{\mathtt{SK}}_i^{(k)}\right\}_{0\le r\le D}, \\ \left\{\widetilde{\mathtt{PK}}_j^{(k)}\right\}_{P_j\in\mathbf{N}_G(i)\cap H, 0\le r\le D-1}\end{array}\right\}_{P_i\in\overline{H}}.$$

Note that the second view can be easily computed by a simulator given the outputs of dishonest parties. It remains to show that those views are computationally indistinguishable. Note that for any $P_j \in \mathbf{N}_G(\overline{H}) \cap H$ the public-key $\mathtt{PK}_j^{(k)}$ has the form $\mathtt{pk}_1 \cdot \mathtt{pk}$ where

$$\mathtt{pk}_1 = \overline{\mathtt{PK}}_j^{(k)} \text{ and } \mathtt{pk} = \prod_{P_i\in\mathbf{N}_G(j)} \mathtt{PK}_i^{(k-1)}.$$

The indistinguishability therefore follows from the unif-KG property of the RR-MHT-PKE scheme.

*Communication Complexity:* The protocol runs for $D$ rounds and in each round $n \cdot d$ public-keys are sent. □

## Information-Transmission Protocol

In this section we present a topology-hiding information-transmission protocol. Here, each party has a message $m_i$ and a public-key $\mathtt{pk}_i$[10] as input. The output of party $P_i$ is a ciphertext $c_i$ under the public key $\mathtt{pk}_i$.

[10]For notational simplicity we use uppercase letters for public-/private-keys which are part of the setup for $\mathcal{N}_{\mathcal{G}}^{d,D}$ and lowercase letters for arbitrary public-/private-keys.

If all parties input the 0-message, $c_i$ is an encryption of 0. Otherwise, $c_i$ is an encryption of a random, non-zero message.

The information-transmission protocol has a recursive structure and is thus parametrized by a level $k$. The protocol requires that parties have generated local pseudonyms. We therefore assume that the parties have access to a setup for topology-hiding threshold encryption over $\mathcal{N}_G^{d,D}$.

**Definition 10.** *A protocol is a* level-$k$ (topology-hiding) secure information-transmission protocol over a network $\mathcal{N}_G^{d,D}$ *if it has the following properties.*

**Setup, Inputs, and Outputs:** *The parties initially hold a setup for topology-hiding threshold encryption over* $\mathcal{N}_G^{d,D}$ (cf. *Definition 8). Each party holds as input a message* $m_i \in \mathcal{M}$ *and a public key* $\boldsymbol{pk}_i \in \mathcal{PK}$ *(not necessarily part of its setup). The output of each party* $P_i$ *is a ciphertext* $c_i \in \mathcal{C}$.

**Correctness:** *With overwhelming probability the output* $c_i$ *is the encryption of message* $s_i$ *under* $\boldsymbol{pk}_i$ *and randomness* $\rho_i$ *(i.e.* $c_i = \mathsf{Enc}(\boldsymbol{pk}_i, s_i; \rho_i)$*) with*

$$s_i = \begin{cases} 0 & \text{if } m_j = 0 \text{ for all } P_j \in \mathbf{N}_G[i]^k \\ x_i & \text{if } m_j \neq 0 \text{ for at least one } P_j \in \mathbf{N}_G[i]^k \end{cases}$$

*where* $x_i \in \mathcal{M} \setminus \{0\}$ *uniform at random.*

**Topology-Hiding Simulation:** *The adversarial view in a real protocol-execution can be simulated with overwhelming probability given the following values*

$$\{\mathbf{N}_G(i), m_i, \boldsymbol{pk}_i, c_i, \nu_i(\cdot)\}_{P_i \in \overline{H}} \cup \{s_i, \rho_i\}_{\mathbf{N}_G[i]^k \subseteq \overline{H}}.$$

*In other words the simulator gets the neighborhood of dishonest parties (in* $\mathcal{N}_G^{d,D}$*), their protocol in- and outputs, and their local pseudonyms from the setup. For any party* $P_i$ *where the whole* $k$*-neighborhood is dishonest the simulator is additionally given the content* $s_i$ *and the randomness* $\rho_i$ *of output* $c_i$.

The simulation property ensures in particular that (a) the adversary does not learn more about the network topology and that (b) the adversary does not learn the content of ciphertext $c_i$ of party $P_i$ unless it corrupts the entire $k$-neighborhood of $P_i$.

**Protocol InfoTransmisson** $\big(k, (m_1, \texttt{pk}_1), \dots, (m_n, \texttt{pk}_n)\big)$

```
Require: Parties have access to an initialized $\mathcal{N}_G^{d,D}$ and have generated
    local pseudonyms.
Input: Each $P_i$ inputs a message $m_i$ and a public key $\texttt{pk}_i$.
 1: if $k = 0$ then
 2:     Each $P_i$ computes $c_i = \mathsf{Enc}(\texttt{pk}_i, 0)$ if $m_i = 0$ or $c_i = \mathsf{Enc}(\texttt{pk}_i, x_i)$
        if $m_i \neq 0$ where $x_i \in \mathcal{M} \setminus \{0\}$ uniform at random.
 3: else
 4:     Each $P_i$ computes $(\widetilde{\texttt{pk}}_i, \texttt{rk}_i) \twoheadleftarrow \mathsf{RandKey}(\texttt{pk}_i)$ and sends $\widetilde{\texttt{pk}}_i$ to
        each $P_j \in \mathbf{N}_G[i]$ which denotes the received key by $\texttt{pk}_{j,\nu_j(i)}$.
 5:     for $l = 0, \dots, d$ do
 6:         The parties compute ciphertexts $(\widetilde{c}_{1,l}, \dots, \widetilde{c}_{n,l})$ by invoking
            $\mathsf{InfoTransmisson}\big(k-1, (m_1, \texttt{pk}_{1,l}), \dots, (m_n, \texttt{pk}_{n,l})\big)$.
 7:     end for
 8:     Each $P_i$ sends $\widetilde{c}_{i,\nu_i(j)}$ to $P_j \in \mathbf{N}_G[i]$.
 9:     Each $P_i$ computes $c_i = \big(\prod_{P_j \in \mathbf{N}_G[i]} \mathsf{DerandCipher}(\texttt{rk}_i, \widetilde{c}_{j,\nu_j(i)})\big)^{r_i}$
        for a uniform random $r_i \in \{1, \dots, |\mathcal{M}| - 1\}$.
10: end if
Output: Each $P_i$ outputs $c_i$.
```

**Lemma 6.** *Given a secure RR-MHT-PKE scheme and for any parameter $0 \leq k \leq D$ with $d^k = \mathrm{poly}(\kappa)$, protocol $\mathsf{InfoTransmisson}\big(k, (m_1, \mathit{pk}_1), \dots, (m_n, \mathit{pk}_n)\big)$ is a secure level-k information-transmission protocol. The protocol communicates at most $(d+1)^k \cdot n \cdot (\log|\mathcal{PK}| + \log|\mathcal{C}|)$ bits within $2k$ rounds.*

*Proof.* (sketch) *Correctness:* For $k = 0$ each party locally computes $c_i$ as specified by the correctness property. The protocol thus achieves correctness perfectly. For $k > 0$ assume that the protocol achieves correctness for $(k-1)$. More precisely, the output of a party $P_j$ for parameter $(k-1)$ is computed perfectly correct if all $(k-1)$-neighbors have input 0. Otherwise, the output of $P_j$ for parameter $(k-1)$ is computed correctly except with error probability $\varepsilon_{k-1}$. First, we consider the case where all parties in the $k$-neighborhood of $P_i$ have input 0. The assumption for $(k-1)$ implies that all $\widetilde{c}_{i,\nu_i(j)}$ contain 0. The properties of the RR-MHT-PKE scheme imply that $s_i = r_i \cdot 0 = 0$. In the second case at least one party in the $k$-neighborhood of $P_i$ has a non-zero input. This implies that at least one $\widetilde{c}_{i,\nu_i(j)}$ contains a uniform random, non-zero message (with error probability of at most $\varepsilon_{k-1}$). The properties of the RR-MHT-PKE thus ensure that $c_i$ contains a uniform random, non-zero message (except with error probability $\varepsilon_k := \varepsilon_{k-1} + \frac{1}{|\mathcal{M}|}$). This implies an overall success probability of at least $1 - (\frac{k \cdot n}{|\mathcal{M}|})$.

*Topology-Hiding Simulation:* To simulate the view of the adversary the simulator is given

$$\big\{\mathbf{N}_G(i), m_i, \mathtt{pk}_i, c_i, \nu_i(\cdot)\big\}_{P_i \in \overline{H}} \cup \big\{s_i, \rho_i\big\}_{\mathbf{N}_G[i]^k \subseteq \overline{H}}.$$

For $k = 0$ those values correspond exactly to the view of the adversary during an actual protocol execution. Simulation is thus easy. For the case $k > 0$ assume that the view of the adversary can be simulated for $k' < k$. The view of the adversary can now be simulated as follows. At the beginning, the simulator generates all public keys and de-randomizers seen by the adversary. For each dishonest $P_i$ the simulator computes $\mathtt{rk}_i, \widetilde{\mathtt{pk}}_i$ using $\mathsf{RandKey}$. For each honest $P_j$ in the neighborhood of $\overline{H}$ the simulator sets $\widetilde{\mathtt{pk}}_j$ to a random public-key using $\mathsf{KeyGen}$. Due to the IND-CKCA property of the RR-MHT-PKE scheme these public keys are indistinguishable from the corresponding public-keys seen by the adversary in an actual protocol-execution. The above values also determine all keys $\mathtt{pk}_{i,\nu_i(j)}$ for $P_i \in \overline{H}$ and $P_j \in \mathbf{N}_G(i)$. Now, we consider the ciphertexts seen by the adversary in the second part of the protocol. In essence the simulator must generate all $\widetilde{c}_{j,\nu_j(i)}$ where $P_i$ and/or $P_j$ are dishonest. If the whole $(k-1)$-neighborhood of $P_j$ is dishonest the simulator must also provide the content and the randomness of $\widetilde{c}_{j,\nu_j(i)}$ which are required for the sub-simulation of the recursive protocol invocations. We recall that $\mathsf{DerandCipher}$ is efficiently invertible if the de-randomizer is known. First, simulator generates a random $r_i \in \{1, \ldots, |\mathcal{M}| - 1\}$ for each dishonest $P_i$. If the whole $k$-neighborhood of $P_i$ is dishonest (i.e., $\mathbf{N}_G[i]^k \subseteq \overline{H}$) the simulator is additionally given $s_i$ and $\rho_i$. This allows the simulator to compute for each neighbor $P_j \in \mathbf{N}_G[i]$ a valid $s_{j,\nu_j(i)}$, randomness $\rho_{j,\nu_j(i)}$, and an encryption $\widetilde{c}_{j,\nu_j(i)} = \mathsf{Enc}(\widetilde{\mathtt{pk}}_i, s_{j,\nu_j(i)}; \rho_{j,\nu_j(i)})$ such that $c_i = \big(\prod_{P_j \in \mathbf{N}_G[i]} \mathsf{DerandCipher}(\mathtt{rk}_i, \widetilde{c}_{j,\nu_j(i)})\big)^{r_i}$. If there exists a honest party in the $k$-neighborhood of $P_i$, the simulator is not given $s_i$ and $\rho_i$. However, in this case there is at least one $P_j$ in $\mathbf{N}_G[i]$ such that $\mathbf{N}_G[j]^{k-1} \nsubseteq \overline{H}$. This allows the simulator to first generate all $s_{j,\nu_j(i)}, \rho_{j,\nu_j(i)}$ and $\widetilde{c}_{j,\nu_j(i)} = \mathsf{Enc}(\widetilde{\mathtt{pk}}_i, s_{j,\nu_j(i)}; \rho_{j,\nu_j(i)})$ where $\mathbf{N}_G[j]^{k-1} \subseteq \overline{H}$. Then it chooses the remaining $\widetilde{c}_{j,\nu_j(i)}$ randomly under the constraint that $c_i = \big(\prod_{P_j \in \mathbf{N}_G[i]} \mathsf{DerandCipher}(\mathtt{rk}_i, \widetilde{c}_{j,\nu_j(i)})\big)^{r_i}$. In a final step the adversary generates for any honest $P_j \in \mathbf{N}_G[i]$ the values $s_{i,\nu_i(j)}, \rho_{i,\nu_i(j)}$ and $\widetilde{c}_{i,\nu_i(j)} = \mathsf{Enc}(\widetilde{\mathtt{pk}}_j, s_{i,\nu_i(j)}; \rho_{i,\nu_i(j)})$. The IND-TCPA property of the RR-MHT-PKE scheme and the correctness property of the protocol ensure that the generated ciphertexts are indistinguishable from the ones seen by the adversary in an actual protocol execution. Now all values required for the simulation of the $d+1$ invocations of $\mathsf{InfoTransmisson}$ with parameter $(k-1)$ are given. The simulator can thus use the sub-simulator to generate the view of the adversary in the middle part of the protocol.

*Communication Complexity:* Let $f(k)$ be the communication complexity of $\mathsf{InfoTransmisson}\big(k,\dots\big)$. Then we have $f(0) = 0$ and $f(k) = d \cdot n \cdot (\log|\mathcal{PK}| + \log|\mathcal{C}|) + (d+1) \cdot f(k-1)$. This results in a communication complexity of at most $(d+1)^k \cdot n \cdot (\log|\mathcal{PK}| + \log|\mathcal{C}|)$ bits. The round complexity follows from the observation that one can invoke the subprotocols $\mathsf{InfoTransmisson}(k-1,\dots)$ in parallel. □

## Decryption Protocol

In this section we describe a distributed decryption protocol which allows each party $P_i$ to decrypt a ciphertext $c_i$ under its shared private-key $\mathtt{SK}_i$ which has been generated by the setup protocol. The decryption protocol consists of two parts. First the parties jointly compute for each ciphertext $c_i$ a decryption-share $\mathtt{x}_i$ under the shared private-key of $P_i$. In a second phase each party $P_i$ can locally decrypt $c_i$ using the decryption share $\mathtt{x}_i$. First, we present a subprotocol which allows to compute the required decryption shares. The key-idea is to use the homomorphic property of decryption-shares which allows a recursive computation. The subprotocol is therefore parametrized by $k$.

**Definition 11.** *A protocol is a* secure level-$k$ (topology-hiding) decryption-share protocol *over a network $\mathcal{N}_G^{d,D}$ if it has the following properties.*

**Setup, Inputs, and Outputs:** *The parties initially hold a setup for topology-hiding threshold encryption over $\mathcal{N}_G^{d,D}$ (*cf.* Definition 8). Each party $P_i$ inputs a ciphertext $c_i \in \mathcal{C}$. The output of party $P_i$ is a decryption share $\mathtt{x}_i \in \mathcal{DS}$.*

**Correctness:** *With overwhelming probability* $x_i = \mathit{ShareDecrypt}(SK_i^{(k)}, c_i)$ *where* $SK_i^{(k)}$ *is the level-k shared private-key of* $P_i$ *from the setup.*

**Topology-Hiding Simulation:** *The adversarial view in a real protocol-execution can be simulated with overwhelming probability given the following values*

$$\{\mathbf{N}_G(i), c_i, x_i, \nu_i(\cdot), \overline{SK}_i^{(0)}, \ldots, \overline{SK}_i^{(k)}\}_{P_i \in \overline{H}}$$

*In other words the simulator gets the neighborhood of dishonest parties (in* $\mathcal{N}_G^{d,D}$*), their protocol in- and outputs, their local pseudonyms, and their private-key shares (up to level-k) of the assumed setup.*

The simulation property ensures in particular that the adversary does not learn more about the network topology.

**Protocol DecShares**$(k, c_1, \ldots, c_n)$

**Require:** Parties have access to an initialized $\mathcal{N}_G^{d,D}$ and have generated a setup for topology-hiding threshold encryption over $\mathcal{N}_G^{d,D}$.
**Input:** Each $P_i$ inputs a ciphertext $c_i$.

```
 1: if $k = 0$ then
 2:     Each $P_i$ computes $x_i = \mathsf{ShareDecrypt}(\overline{SK}_i^{(0)}, c_i)$.
 3: else
 4:     Each $P_i$ computes $(rk_i, \widehat{c_i}) = \mathsf{RandCipher}(c_i)$ and sends $\widehat{c_i}$ to each
        $P_j \in \mathbf{N}_G(i)$ which denotes the received value by $c_{j,\nu_j(i)}$.
 5:     for $l = 1, \ldots, d$ do
 6:         Parties jointly compute $(x_{1,l}, \ldots, x_{n,l}) = \mathsf{DecShares}(k - 1, c_{1,l}, \ldots, c_{n,l})$.
 7:     end for
 8:     Each $P_i$ sends $x_{i,\nu_i(j)}$ to each $P_j \in \mathbf{N}_G(i)$.
 9:     Each $P_i$ computes first $\widehat{x}_i = \prod_{P_j \in \mathbf{N}_G(i)} x_{j,\nu_j(i)}$ and then $x_i =$
        $\mathsf{DerandShare}(rk_i, \widehat{x}_i) \cdot \mathsf{ShareDecrypt}(\overline{SK}_i^{(k)}, c_i)$.
10: end if
```

**Output:** Each $P_i$ outputs $x_i$.

**Lemma 7.** *Given a secure RR-MHT-PKE scheme and for any parameter* $0 \leq k \leq D$ *with* $d^k = \mathrm{poly}(\kappa)$ *the above protocol* $\mathit{DecShares}(k, c_1, \ldots, c_n)$

*is a secure level-k decryption-share protocol. The protocol communicates* $d^k \cdot n \cdot (\log|\mathcal{DS}| + \log|\mathcal{C}|)$ *bits within* $2k$ *rounds.*

*Proof.* (sketch) *Correctness:* The correctness essentially follows from the structure of the assumed setup and from the properties of the RR-MHT-PKE scheme. In the case $k = 0$ we have $\mathtt{SK}_i^{(0)} = \overline{\mathtt{SK}}_i^0$ which implies $\mathtt{x}_i = \mathsf{ShareDecrypt}(\mathtt{SK}_i^{(0)}, c_i)$. For $k > 0$ we have $\mathtt{SK}_i^{(k)} = \overline{\mathtt{SK}}_i^{(k)} + \sum_{P_j \in \mathbf{N}_G(i)} \mathtt{SK}_j^{(k-1)}$. The properties of the RR-MHT-PKE scheme thus imply that $\mathtt{x}_i = \mathsf{ShareDecrypt}(\mathtt{SK}_i^{(k)}, c_i)$ (c.f. protocol line 9).

*Topology-Hiding Simulation:* In the case $k = 0$ the view of the adversary is directly determined by values given to the simulator. Simulation is therefore easy to achieve. In the case $k > 0$ the simulation of the adversarial view works similar as for the information-transmission protocol (we recall that $\mathsf{DerandShare}$ is efficiently invertible if the de-randomizer is known). The simulator essentially emulates the protocol run. The IND-CKCA property of the RR-MHT-PKE scheme allows the simulator to choose random ciphertexts for $c_{i,\nu_i(j)}$ of honest $P_j$. Moreover, the decryption shares $\mathtt{x}_{j,\nu_j(i)}$ for honest $P_j$ can also be chosen randomly (where the distribution is conditioned on the outputs of dishonest parties). The view during the executions of $\mathsf{DecShares}$ with parameter $k-1$ can be generated using the $(k-1)$-subsimulator guaranteed by the induction hypothesis.

*Communication Complexity:* Denote by $f(k)$ be the communication complexity of $\mathsf{DecShares}(k, \dots)$. Then we have $f(0) = 0$ and $f(k) = n \cdot d \cdot (\log|\mathcal{DS}| + \log|\mathcal{C}|) + d \cdot f(k-1)$. This results in a communication complexity of $f(k) = d^k \cdot n \cdot (\log|\mathcal{DS}| + \log|\mathcal{C}|)$. The round complexity follows from the observation that one can invoke the subprotocols $\mathsf{DecShares}(k-1, \dots)$ in parallel. □

**Definition 12.** *A protocol is a* secure (topology-hiding) threshold decryption protocol *for network* $\mathcal{N}_\mathcal{G}^{d,D}$ *if it has the following properties.*

**Setup, Inputs and Outputs:** *The parties initially hold a setup for topology-hiding threshold encryption over* $\mathcal{N}_\mathcal{G}^{d,D}$ *(*cf. *Definition 8). Each party* $P_i$ *inputs a ciphertext* $c_i \in \mathcal{C}$*. The output of party* $P_i$ *is a message* $m_i$*.*

**Correctness:** *With overwhelming probability it holds for each party* $P_i$

*that* $m_i = \mathsf{Decode}(\mathsf{ShareDecrypt}(\mathsf{SK}_i, c_i))$ *where* $\mathsf{SK}_i$ *is the shared private-key of* $P_i$*.*

**Topology-Hiding Simulation:** *The adversarial view in a real-protocol execution can be simulated with overwhelming probability given the following values*

$$\{\mathbf{N}_G(i), c_i, m_i, \nu_i(\cdot), \overline{\mathsf{SK}}_i^{(0)}, \ldots, \overline{\mathsf{SK}}_i^{(D)}\}_{P_i \in \overline{H}}$$

*In other words the simulator gets the neighborhood of dishonest parties (in* $\mathcal{N}_{\mathcal{G}}^{d,D}$*), their protocol in- and outputs, their local pseudonyms, and their private-key shares of the assumed setup.*

**Protocol Decryption**$(c_1, \ldots, c_n)$

```
Require: Parties have access to an initialized $\mathcal{N}_{\mathcal{G}}^{d,D}$ and have generated
    a setup for topology-hiding threshold encryption over $\mathcal{N}_{\mathcal{G}}^{d,D}$.
Input: Each $P_i$ inputs a ciphertext $c_i$.
 1: The parties compute $(\mathtt{x}_1, \ldots, \mathtt{x}_n) = \mathsf{DecShares}(D, c_1, \ldots, c_n)$.
Output: Each $P_i$ outputs $\mathsf{Decode}(\mathtt{x}_i, c_i)$.
```

**Lemma 8.** $\mathsf{Decryption}(k, c_1, \ldots, c_n)$ *is a secure threshold decryption protocol given a secure RR-MHT-PKE scheme The protocol communicates* $d^D \cdot n \cdot (\log|\mathcal{DS}| + \log|\mathcal{C}|)$ *bits within* $2D$ *rounds.*

*Proof.* (sketch) *Correctness:* The correctness follows from Lemma 7 and the properties of the RR-MHT-PKE scheme.

*Topology-Hiding Simulation:* The adversarial view in a real protocol execution can be simulated as follows (recall that $\mathsf{Decode}$ is efficiently invertible). First the simulator computes for each pair $(c_i, m_i)$ a decryption share $\mathtt{x}_i$ such that $m_i = \mathsf{Decode}(\mathtt{x}_i, c_i)$. The rest of the view can then be generated using the sub-simulator for $\mathsf{DecShares}(D, \ldots)$.

*Communication Complexity:* The communication complexity and the number of rounds follows directly from the invocation of $\mathsf{DecShares}$ with parameter $D$. $\square$

### 3.4.2 Multi-Party Boolean OR

In this section we present a protocol which securely and topology-hidingly realizes the multi-party Boolean-OR functionality $\mathcal{F}_{\mathrm{OR}}$ using the topology-hiding threshold encryption protocol from the previous section. The functionality $\mathcal{F}_{\mathrm{OR}}$ takes from each party $P_i$ an input bit $b_i$ and computes the OR of those bit, i.e., $b = b_1 \vee \cdots \vee b_n$.

**Functionality $\mathcal{F}_{\mathrm{OR}}$**

1. Every party $P_i$ (and the adversary on behalf of corrupted parties) sends (input) bit $b_i$; if $P_i$ does not submit a valid input, then $\mathcal{F}_{\mathrm{OR}}$ adopts $b_i = 0$.
2. Every party $P_i$ receives (output) $b = b_1 \vee \cdots \vee b_n$.

**Assumptions.** We assume in the following that the parties have access to a secure RR-MHT-PKE scheme with security parameter $\kappa$, where $n = \mathrm{poly}(\kappa)$. Moreover, parties are given the network $\mathcal{N}_{\mathcal{G}}^{d,D}$ where the graphs in the family $\mathcal{G}$ are connected, have a maximum degree of $d \leq n$, and a diameter of at most $D \leq n$ where $d$ and $D$ are publicly known.

**Protocol Boolean-OR$(b_1, \ldots, b_n)$**

**Initialization:**

```
1: Each party $P_i$ inputs its neighborhood $\mathbf{N}_G[i]$ into $\mathcal{N}_{\mathcal{G}}^{d,D}$.
2: The parties generate a setup for topology-hiding threshold
   encryption over $\mathcal{N}_{\mathcal{G}}^{d,D}$ using GenerateSetup.
```

**Computation:**

```
Input: Each party $P_i$ inputs a bit $b_i$.
1: Each party $P_i$ sets $m_i = 0$ if $b_i = 0$. Otherwise, its sets $m_i$ to an
   arbitrary message in $\mathcal{M} \setminus \{0\}$.
2: The parties compute
   $(c_1, \ldots, c_n) = \mathsf{InfoTransmisson}\big(D, (m_1, \mathtt{PK}_1), \ldots, (m_n, \mathtt{PK}_n)\big)$.
3: The parties compute $(m'_1, \ldots, m'_n) = \mathsf{Decryption}(c_1, \ldots, c_n)$.
Output: If $m'_i = 0$ $P_i$ outputs 0. Otherwise it outputs 1.
```

**Lemma 9.** *Given a secure RR-MHT-PKE scheme and for $d, D$ with $d^D = \text{poly}(\kappa)$ the protocol $\mathsf{Boolean\text{-}OR}(b_1, \dots, b_n)$ securely and topology-hidingly realizes $\mathcal{F}_{\text{OR}}$ (in the $\mathcal{N}_{\mathcal{G}}^{d,D}$-hybrid model). In the initialization phase the protocol $\mathsf{Boolean\text{-}OR}(b_1, \dots, b_n)$ communicates $D \cdot d \cdot n \cdot \log|\mathcal{PK}|$ bits within $D$ rounds. In the computation phase the protocol communicates at most $(d+1)^D \cdot n \cdot (\log|\mathcal{DS}| + \log|\mathcal{PK}| + 2\log|\mathcal{C}|)$ bits within $4 \cdot D$ rounds.*

*Proof. Correctness:* We assume the condition $d^D = \text{poly}(\kappa)$. The correctness thus follows directly from the properties of Lemmas 5, 6, and 8 as the information-transmission protocol essentially allows to compute Boolean-ORs.

*Topology-Hiding Simulation:* Given the values $\{\mathbf{N}_G(i), b_i, b\}_{P_i \in \overline{H}}$ the view of the adversary can be simulated as follows. First the simulator generates a setup for $\mathcal{N}_{\mathcal{G}}^{d,D}$. Next, for each dishonest $P_i$ the simulator computes the messages $m_i$ and $m_i'$. It generates the corresponding ciphertext $c_i$ (including the randomness). With those values the simulator now runs the sub-simulators for $\mathsf{GenerateSetup}$, $\mathsf{InfoTransmisson}(D, \dots)$, and $\mathsf{Decryption}(\dots)$. The properties of Lemmas 5, 6, and 8 ensure that the generated view is indistinguishable (for the adversary) from a real protocol execution.

*Communication Complexity:* The claimed communication complexity follows directly from the used subprotocols. □

**Remark.** *If the RR-MHT-PKE is instantiated using the DDH based construction from Section 3.3.3, the computation complexity of the protocol $\mathsf{Boolean\text{-}OR}$ is similar to its communication complexity.*

### 3.4.3 Topology-Hiding Broadcast

In this section we describe a protocol which securely realizes the (bit) broadcast functionality $\mathcal{F}_{\text{BC}}^s$, while making black-box use of the $\mathcal{F}_{\text{OR}}$ functionality from the previous section. The functionality $\mathcal{F}_{\text{BC}}^s$ allows sender $P_s$ to input a bit $b_s$ which is output to all parties. This result directly implies that one can securely and topology-hidingly realize secure channels and broadcast using black-box RR-MHT-PKE.

```
Protocol Broadcast($P_s, b_s$)
Require: The sender $P_s$ inputs a bit $b_s$.
 1: The parties compute $(b, \dots, b) = \mathcal{F}_{\text{OR}}(0, \dots, b_s, \dots, 0)$.
Output: Each party $P_i$ outputs $b$.
```

**Lemma 10.** *The protocol Broadcast$(P_s, b_s)$ securely realizes the $\mathcal{F}_{\text{BC}}^s$ functionality in the $\mathcal{F}_{\text{OR}}$-hybrid model.*

*Proof.* We have that $b = 0 \vee \cdots \vee b_s \vee \cdots \vee 0 = b_s$ which implies correctness. The view of the adversary in an actual protocol execution consists of inputs and outputs of dishonest parties and is therefore easy to simulate. $\square$

This implies the following Corollary.

**Corollary 1.** *For $d, D$ with $d^D = \text{poly}(\kappa)$ one can securely and topology-hidingly realize $\mathcal{F}_{\text{BC}}^s$ (in the $\mathcal{N}_{\mathcal{G}}^{d,D}$-hybrid model) given a secure black-box RR-MHT-PKE scheme while communicating at most $(d+1)^D \cdot n \cdot (\log|\mathcal{DS}| + \log|\mathcal{PK}| + 2\log|\mathcal{C}|) + D \cdot d \cdot n \cdot \log|\mathcal{PK}|$ bits within $5 \cdot D$ rounds per invocation.*

Moreover, parties can simply realize secure channels given broadcast. First the receiver generates a key pair and broadcasts the public-key. The sender then broadcasts his message encrypted under this public-key.

**Corollary 2.** *For $d, D$ with $d^D = \text{poly}(\kappa)$ one can securely and topology-hidingly realize secure channels (in the $\mathcal{N}_{\mathcal{G}}^{d,D}$-hybrid model) using black-box RR-MHT-PKE. The communication complexity is twice the one of the broadcast protocol.*

## 3.5 Applications

In this section we provide two applications of our network-hiding communication protocols. Namely, one can securely and topology-hidingly realize MPC and anonymous broadcast.

### 3.5.1 Topology-Hiding Secure MPC

The protocols from the previous section allow parties to topology-hidingly realize a complete network of secure channels (including broadcast channels). They can then use this network to execute a multi-party protocol of their choice, e.g., [GMW87, Pas04]. This easily proves the following result.

**Theorem 2.** *For $d, D$ with $d^D = \mathrm{poly}(\kappa)$ one can securely and topology-hidingly realize any given multi-party functionality (in the $\mathcal{N}_{\mathcal{G}}^{d,D}$-hybrid model) using black-box RR-MHT-PKE.*

*Proof.* This directly follows from Corollaries 1 and 2. □

### 3.5.2 Anonymous Broadcast

Theorem 2 implies that one can topology-hidingly realize anonymous channels given black-box access to a RR-MHT-PKE scheme. But using generic MPC to achieve an anonymous channel is expensive in terms of communication complexity. We therefore provide a protocol in the $\mathcal{F}_{\mathrm{OR}}$-hybrid model which directly realizes *anonymous broadcast* $\mathcal{F}_{\mathrm{ABC}}$.

The functionality $\mathcal{F}_{\mathrm{ABC}}$ generates for each party a unique but random pseudonym. In the subsequent communication rounds each party can publish messages under its pseudonym. Message are linkable which means that parties can relate messages to pseudonyms. If desired, parties can prevent this by generating fresh pseudonyms (e.g., after each communication round).

**Functionality $\mathcal{F}_{\mathrm{ABC}}$**

**Initialization:**

```
1: The functionality generates a random permutation $\sigma$ of $n$ elements.
2: Each party $P_i$ gets output $\sigma(i)$.
```

**Communication Step:**

```
Require: Each party $P_i$ inputs a bit $b_i$.
Output: The parties get the vector $(o_1, \ldots, o_n)$ as output where
    $o_{\sigma(i)} = b_i$.
```

**Anonymous Broadcast Protocol.** The high-level idea of our construction is as follows. In a scheduling phase each party gets a random (but unique) communication slot $\sigma(i)$ assigned. In a communication round for each slot $\sigma(i)$ the $\mathcal{F}_{\mathrm{OR}}$ functionality is invoked which allows $P_i$ to broadcast its bit.

The major challenge is to compute the slot assignment. We solve this issue with a scheduling loop[11]. At the beginning each party selects a random slot. Then over several scheduling rounds the parties resolve colliding selections by computing a reservation matrix. The size of this matrix (parametrized by $m$) determines the collision detection probability. A larger $m$ means a faster expected run time at the cost of increased communication costs per round.

**Protocol AssignSlots**$(m)$

```
1: Each party $P_i$ chooses a random slot $s_i \in \{1, \dots, n\}$.
2: repeat
3:     Each party $P_i$ chooses a random token $r_i \in \{1, \dots, m\}$ and computes the $n \times m$-matrix $A^{(i)} = (a^{(i)}_{x,y})$ where $a^{(i)}_{s_i,r_i} = 1$ and $a^{(i)}_{x,y} = 0$ otherwise.
4:     The parties compute the matrix $A = (a_{x,y})$ where $a_{x,y} = a^{(1)}_{x,y} \vee \dots \vee a^{(n)}_{x,y}$ by invoking $\mathcal{F}_{\mathrm{OR}}$.
5:     If there exists an $r < r_i$ such that $a_{s_i,r} = 1$ party $P_i$ chooses a new random slot $s_i \in \{1, \dots, n\}$ such that $s_i$-th row of $A$ contains only zeros.
6: until Each row of $A$ contains exactly one 1.
Output: Each party $P_i$ outputs $s_i$.
```

**Lemma 11.** *The protocol AssignSlots$(m)$ for the $\mathcal{F}_{\mathrm{OR}}$-hybrid model securely computes a random permutation $\sigma$ of $n$ elements where each party $P_i$ learns $\sigma(i)$. The expected number of rounds the protocol requires to compute the permutation is bounded by $\frac{m}{m-1} \cdot n$ where $\mathcal{F}_{\mathrm{OR}}$ is invoked $n \cdot m$ times per round.*

*Proof. Correctness:* The protocol terminates if each row of $A$ contains exactly one non-zero entry. Thus each slot in $\{1, \dots, n\}$ has been chosen at least by one party. As there are $n$ parties this also means that no slot was chosen twice. The output is therefore a valid permutation. Inspection of the protocol also reveals that the permutation is chosen uniform at

[11] A similar idea was used recently in [KNS16].

random (we consider passive security).

*Termination:* Next, we show that the protocol eventually terminates. Each slot is in one of three states. Either its empty, or its selected by multiple parties, or it is assigned to a single party. We observe that the state transition function for slots is monotone. A selected slot cannot become empty and an assigned slot stays assigned to the same party. In each round where a collision is detected at least one empty slot becomes assigned. After at most $n$ such rounds there are no empty slots left. But this also means that each slot is selected by at least one party and the protocol terminates.

*Round Complexity:* The above argument also leads to a crude upper bound on the number of expected rounds. We observe that a collision between two parties is detected with a probability of at least $p = (1 - \frac{1}{m})$. The expected number of rounds required to detect a collision is therefore at most $\frac{1}{p} = \frac{m}{m-1}$ (geometric distribution). The number of expected rounds is thus bounded by $\frac{m}{m-1} \cdot n$.

*Simulation:* It remains to consider the simulation of the adversarial view. We observe that the (current) slot selection of dishonest parties is enough to simulate the view of the adversary in a scheduling round. The simulator can therefore essentially emulate the protocol (conditioned on the final slots of dishonest parties). □

We can now combine this scheduling protocol with the broadcast protocol from Section 3.4.3 to build a protocol which achieves anonymous broadcast.

**Protocol Anonymous Broadcast**$(m)$

**Initialization:**

```
1: The parties compute $(\sigma(1), \dots, \sigma(n)) = \mathsf{AssignSlots}(m)$.
```

**Communication Step:**

```
Require: Each party $P_i$ inputs a bit $b_i$.
1: for $s = 1, \dots, n$ do
2:     The parties compute
       $(o_s, \dots, o_s) = \mathsf{Boolean\text{-}OR}(0, \dots, b_{\sigma^{-1}(s)}, \dots, 0)$.
3: end for
```

**Output:** Each party $P_i$ outputs vector $(o_1, \ldots, o_n)$.

**Lemma 12.** *The protocol* $\mathsf{Anonymous\ Broadcast}(m)$ *securely realizes the functionality* $\mathcal{F}_{\mathrm{ABC}}$ *in the* $\mathcal{F}_{\mathrm{OR}}$*-hybrid model.*

*Proof.* The statement follows directly from Lemmas 11 and 10. □

**Corollary 3.** *For* $d, D$ *with* $d^D = \mathrm{poly}(\kappa)$ *one can securely and topology-hidingly realize* $\mathcal{F}_{\mathrm{ABC}}$ *(in the* $\mathcal{N}_{\mathcal{G}}^{d,D}$*-hybrid model) given a secure black-box RR-MHT-PKE scheme*

# Chapter 4

# Classification of Consistency Specifications

The content of this chapter is based on the works [LMT16] and [LMT17].

## 4.1 Introduction

The seminal result of [LSP82] and [KY84] states that given authenticated channels, broadcast can be achieved if and only if strictly less than $\frac{n}{3}$ of the involved parties behave dishonestly, even if an error probability of less than $\frac{1}{3}$ were tolerated. This raises questions such as "What is required to construct broadcast if more parties are dishonest?" and "What are the consistency guarantees one can achieve with authenticated channels?". Broadcast itself guarantees a very strong form of consistency. The study of primitives with a weaker form of consistency guarantee is well-motivated for two different reasons described below.

First, as argued by Lamport in [Lam83], there are settings of practical relevance where a weaker form of broadcast is sufficient. Specifically, in the *transaction commit problem*, a database transaction is coordinated by some (not necessarily honest) party $P_1$ who decides whether a transaction should be committed or aborted. A single dishonest party $P_i$ may be enough to cause the transaction to be aborted, but in this case, the honest parties must agree on whether to abort the transaction, or to commit to it. To formalize this setting, [Lam83] introduced a weaker form of broadcast, which we will henceforth refer to as a *weak broadcast channel.* This channel behaves like a regular broadcast channel if all parties are honest, but requires the validity condition to hold *only if every party is honest.* Such a guarantee may be achievable even if broadcast is not achievable.

Second, such a weaker primitive might be assumed to be available, and one can ask whether a stronger primitive (e.g. a broadcast channel) can be achieved by a protocol that not only can use authenticated channels, but also has access to the weaker primitive. A result of this type, proved in [FM00], is that broadcast is achievable up to $\frac{n}{2}$ cheaters, assuming that each party can broadcast to any two other parties.

The ultimate goal of a theory in this field is a characterization of various levels of consistency guarantees as well as the hierarchy between them.

### 4.1.1 Contribution and Outline

In this chapter, we are concerned with refining the hierarchy between different types of consistency guarantees.

We proceed as follows. In Section 4.2, we revisit the notion of consistency specifications introduced in [Mau04]. A consistency specification captures, for every set $H$ of (assumed) honest parties and for every tuple of input values of these honest parties, which tuples of output values are possible, no matter what the other parties do. In other words, a specification guarantees that no adversarial behavior can result in the honest parties' output values to be outside the specified set of tuples. Note that while this concept captures consistency guarantees in the most general form, it (intentionally) does not capture secrecy guarantees. In this section we provide rigorous definitions of the basic concepts and introduce different flavors of constructions. We also extend the framework of [Mau04] to include probabilistic protocols (rather than only deterministic ones).

Next in Section 4.3, we investigate a stronger form of the broadcast impossibility result shown in [LSP82]. As it is common for impossibility results in distributed computing, we show all of our results in the setting of three parties. We prove that even if two of the three parties can broadcast values, there is no protocol that would allow the third party to broadcast a value. The proof of this result requires a generalized version of so called “scenario”-proofs (see, e.g., [FLM85]) where additional primitives are given. This contribution, which is used throughout the chapter, is of independent interest beyond the specific results of this work.

Section 4.4 deals with the classification of consistency specifications. Here, one considers the closure of a given a set of consistency specifications, i.e., all consistency specifications which one can construct from this set. This leads to a natural classification where two specifications are in the same class if they have the same closure. As an example we give a complete classification of three-party specifications where a fixed party can give a binary input and the other two parties each have a binary output.

In Section 4.5 we investigate the hierarchy of consistency primitives between authenticated channels and broadcast. To this end, we propose an intermediate level specification for three-parties which we call XOR-cast. This channel takes a bit $b_i$ from $P_i$ and a bit $b_j$ from $P_j$ as input. If all parties behave correctly, the value of $b_i \oplus b_j$ should be output by all parties. If one of the parties $P_i$ or $P_j$ is dishonest, the honest parties must output the same value. If the third $P_k$ is dishonest, the remaining parties must output $b_i \oplus b_j$. We show a strong separation between authenticated channels and broadcast by proving two strong impossibility results, where we call an impossibility *strong* if it holds even if a constant error probability is tolerated and even if an arbitrary number of communication rounds are allowed. First, it is strongly impossible to achieve XOR-cast from authenticated communication. Second, it is strongly impossible to achieve broadcast from XOR-cast and authenticated communication. This demonstrates that the hierarchy of primitives has a more complex structure than previously known.

### 4.1.2 Related Work

Results on the possibility and impossibility of achieving broadcast when other primitives (stronger than authenticated communication) are available were proved in [BGP89, PW96, CFF$^+$05, HMR14, Ray15]. In a related line of work, [JMS12, RMS$^+$04] derive combinatorial lower bounds on the number of partial broadcast channels among a set of parties needed in order to still be able to achieve broadcast. The general problem of constructing consistency primitives from assumed such primitives was proposed and formalized in [Mau04]. The notion of consistency specifications does not allow to model secrecy requirements. A consistency specification thus corresponds to a trusted party which enables dishonest parties to learn the inputs of honest parties. For a more general setting where secrecy matters, one may use security frameworks such as the universal composition framework [Can01] or the constructive cryptography framework presented in [MR11] and [Mau11].

In [Lam83, FLM85] it is shown that there exists no perfectly secure protocol which constructs weak broadcast from authenticated channels in a finite number of rounds if $\frac{n}{3}$ or more of the parties behave dishonestly. On the other hand, Lamport provides a protocol which achieves weak broadcast, but requires an infinite amount of runtime. This suggests that weak broadcast is in some sense weaker than broadcast; namely, the result in [LSP82] implies that there exists no such approximation protocol for broadcast. However, in distributed computing or MPC one is mostly interested in protocols which run for a fixed number of rounds (or at least terminate eventually). Here, Lamport's results show that both weak broadcast and broadcast cannot be achieved with zero error probability given authenticated channels. If one allows protocols with an error probability negligible in the number of rounds, the impossibility for broadcast still holds. On the other hand, it was shown in [FGH$^+$02] that weak broadcast can be achieved from authenticated channels with arbitrary small error probability. Moreover, [LSP82, Lam83, FGH$^+$02] do not consider the relation between weak broadcast and broadcast. Especially, it is not shown whether broadcast can be achieved given weak broadcast.

Upper bounds on the success probability for probabilistic broadcast and Byzantine agreement were also studied in [KY84, GY89]. The work of Karlin and Yao [KY84] gives an upper bound of $\frac{2}{3}$ for the fully synchronous, round-based setting. Somewhat surprisingly, the work of Graham and Yao [GY89] considers a synchronous model with a rushing adversary that can observe the inputs of all other parties in each round before deciding on its own input for the round. In this setting, [GY89] show the stronger bound of $(\sqrt{5}-1)/2$ and also give protocols that match this bound. Such a stronger bound is possible only because the guarantee is stronger and includes a secrecy guarantee: the adversary must not learn the output too early.

## 4.2 Preliminaries

In this chapter we assume that the honest parties in $\mathcal{P}$ will execute protocol instructions whereas dishonest parties can deviate arbitrarily from the protocol.

### 4.2.1 Consistency Specifications

Primitives, such as a broadcast channel, provide the honest parties with consistency guarantees. That is, for every set $H$ of honest parties and every possible choice $\vec{x}_H$ of their inputs the consistency guarantees restrict the set of possible outputs of the honest parties. Consistency guarantees limit the influence of dishonest parties on the possible outputs of honest parties. We thus model such primitives as functions called *consistency specifications* that map a set of honest parties along with their inputs, to a non-empty set of possible outputs of those parties. Here, a smaller set of potential outputs implies a stronger guarantee offered by the consistency specification, since the uncertainty over the actual output is smaller. More formally, a consistency specification (introduced in [Mau04]) with input domain $\mathcal{D}$ and output domain $\mathcal{R}$ is defined as follows.

**Definition 13.** *A* consistency specification *with input domain $\mathcal{D}$ and output domain $\mathcal{R}$ is a function which assigns every non-empty subset $H \subseteq \mathcal{P}$ and every input tuple $\vec{x}_H \in \mathcal{D}^H$ a non-empty set $\mathcal{C}(H, \vec{x}_H) \subseteq \mathcal{R}^H$ of output tuples. It satisfies the following* monotonicity constraint*: For any non-empty subsets $H' \subseteq H \subseteq \mathcal{P}$*

$$\mathcal{C}(H, \vec{x}_H)|_{H'} \subseteq \mathcal{C}(H', \vec{x}_{H|H'}). \tag{4.1}$$

The monotonicity constraint models the fact that dishonest parties can in particular behave correctly. This implies that larger sets of honest parties cannot have weaker consistency guarantees.

**Remark.** *Our definition of a consistency specification differs slightly from the original version in [Mau04] where parties may have different input (or output) domains. Here, parties are not allowed to have different input (or output) domains. However, this is not a limitation. If parties should have different input (resp. output) domains the actual input (resp. output) domain of the consistency specification can be set to the union of all those* *domains and inputs outside a party's input domain can be mapped to a default input. Moreover, if all parties are dishonest, i.e., $H = \emptyset$, there are no consistency guarantees to be formulated. We therefore restrict the domain of consistency specifications to non-empty subsets $H \subseteq \mathcal{P}$.*

**Examples of Consistency Specifications.** We consider two important examples of consistency specifications that we will use throughout this chapter. In both examples the input and output domain are bits, i.e., $\mathcal{D} = \mathcal{R} = \{0, 1\}$.

**Example 1.** *A bit* broadcast channel $\mathrm{BC}_i$ *for sender $P_i$ is the following consistency specification:*

$$\mathrm{BC}_i(H, \vec{x}_H) = \left\{ \vec{y}_H \in \{0,1\}^H \;\middle|\; \begin{array}{l} \exists v \; \big( \; (\forall j \in H : \vec{y}_{H|\{j\}} = v) \\ \wedge \; (i \in H \Rightarrow v = \vec{x}_{H|\{i\}}) \; \big) \end{array} \right\}.$$

**Example 2.** *An* authenticated bit-channel $\textsc{Auth}_{i,j}$ *from $P_i$ to $P_j$ guarantees that $P_j$'s output is equal to the input of $P_i$ if both of them are honest:*

$$\textsc{Auth}_{i,j}(H, x_H) = \left\{ \vec{y}_H \in \{0,1\}^H \;\middle|\; i, j \in H \Rightarrow \vec{y}_{H|\{j\}} = \vec{x}_{H|\{i\}} \right\}.$$

**Parties without Input or Output.** Formally, a consistency specification requires that any (honest) party has (to provide) an input and has (to receive) an output. However, in the above example inputs of all (honest) parties except $P_i$ have no influence on the consistency guarantee. We say that such parties have no input. Similarly, for the authenticated channel $\textsc{Auth}_{i,j}$ the output of all parties except $P_j$ provide no information (they are arbitrary). We say that such parties have no output.

**Definition 14.** *Let $\mathcal{C}$ be a consistency specification with input domain $\mathcal{D}$ and output domain $\mathcal{R}$. A party $P_i$ has* no input *if for every $H$ with $P_i \in H$ and all $\vec{a}_H, \vec{b}_H \in \mathcal{D}^H$ with $\vec{a}_H|_{H\setminus\{i\}} = \vec{b}_H|_{H\setminus\{i\}}$ it holds that $\mathcal{C}(H, \vec{a}_H) = \mathcal{C}(H, \vec{b}_H)$. A party $P_i$ has* no output *if for every $H$ with $P_i \in H$ and all $\vec{x}_H$ it holds that $\mathcal{C}(H, \vec{x}_H)|_{\{i\}} = \mathcal{R}$.*

For simplicity, we omit the (formal) input (resp. output) of parties with no input (resp. no output). For instance, in the broadcast specification $\mathrm{BC}_i$ we only give the input of (honest) $P_i$.

**Parallel Composition.** The parallel composition of consistency specifications allows to model the situation where parties have access to multiple specifications at once. More formally, consider consistency specifications $\mathcal{C}^{(1)}, \dots, \mathcal{C}^{(\ell)}$ with input domain $\mathcal{D}$ and output domain $\mathcal{R}$.

**Definition 15.** *The* parallel composition of $\mathcal{C}^{(1)}, \dots, \mathcal{C}^{(\ell)}$ *is consistency specification* $[\mathcal{C}^{(1)}, \dots, \mathcal{C}^{(\ell)}]$ *with input domain* $\mathcal{D}^\ell$ *and output domain* $\mathcal{R}^\ell$ *where for every* $H \subseteq \mathcal{P}$ *and all* $\vec{x}_H = \big((x_{ij})_{j\in[\ell]}\big)_{i\in H} \in \mathcal{D}$ *it holds that*

$$\mathcal{C}(H, \vec{x}_H) = \left\{ \vec{y}_H \in \mathcal{R}^{\ell H} \;\middle|\; \begin{array}{l} \vec{y}_H = \big((y_{ij})_{j\in[\ell]}\big)_{i\in H} \\ \wedge\ \forall j\ (y_{ij})_{i\in H} \in \mathcal{C}^{(j)}\big(H, (x_{ij})_{i\in H}\big) \end{array} \right\}.$$

**Example 3.** *The* parallel composition *of multiple authenticated channels* $\textsc{Auth}_{i,j}$ *for all* $i, j \in \mathcal{P}$ *forms the complete network* $\textsc{Auth}$.

### 4.2.2 Protocols and Constructions

A protocol allows the parties to construct a new consistency specification. In each protocol round parties (locally) compute inputs a consistency specification using their protocol input and outputs from previously invoked consistency specifications. At the end of the protocol execution each party computes its protocol output as a function of its protocol input and all the outputs it received from invoked specifications.

**Deterministic Protocols.** A deterministic protocols runs for $\ell \geq 0$ rounds. In each round $r$, party $P_i$ uses the deterministic round function $f_i^{(r)}$ to compute its input for the round specification $\mathcal{C}^{(r)}$ which has input domain $\mathcal{D}_r$ and output domain $\mathcal{R}_r$. At the end of the last round, party $P_i$ uses its output function $g_i$ to compute its protocol output. Denote by $\vec{\mathcal{C}} = (\mathcal{C}^{(r)})_{r\in 1,\dots,\ell}$ the tuple of invoked specifications. Then we can define a deterministic protocol as follows.

**Definition 16** ([Mau04])**.** *A deterministic* $\ell$-round protocol $\Pi$ *for tuple* $\vec{\mathcal{C}}$ *with input domains* $\mathcal{D}$ *and output domains* $\mathcal{R}$ *consists of round functions*

$$f_i^{(r)} : \mathcal{D} \times \mathcal{R}_1 \times \cdots \times \mathcal{R}_{r-1} \to \mathcal{D}_r \quad \forall i \in \mathcal{P}\ \forall r \in [\ell]$$

*and output functions*

$$g_i : \mathcal{D} \times \mathcal{R}_1 \times \cdots \times \mathcal{R}_\ell \to \mathcal{R} \quad \forall i \in \mathcal{P}.$$

We explicitly allow zero-round protocols where no consistency specifications are invoked. By executing the protocol $\Pi$ using tuple $\vec{\mathcal{C}}$, the parties achieve a new consistency specification denoted by $\Pi\vec{\mathcal{C}}$. More formally, the output of $\Pi\vec{\mathcal{C}}$ is computed by iteratively applying the round functions of $\Pi$ to the input tuple $\vec{x}_H$.

**Definition 17.** *For a protocol* $\Pi$ *and the corresponding tuple* $\vec{\mathcal{C}}$ *the* protocol specification $\Pi\vec{\mathcal{C}}$ *is the following consistency specification, such that for every* $H \subseteq \mathcal{P}$ *and* $\vec{x}_H = (x_i)_{i\in H} \in \mathcal{D}^H$

$$\Pi\vec{\mathcal{C}}(H,\vec{x}_H) = \left\{ (y_i)_{i\in H} \in \mathcal{R}^H \;\middle|\; \begin{array}{l} \forall r \in [\ell]\ \exists (x_{ir})_{i\in H} \in \mathcal{D}_r^H\ \exists (y_{ir})_{i\in H} \in \mathcal{R}_r^H \\ \forall i \in H\text{:}\quad x_{ir} = f_i^{(r)}(x_i, y_{i1}, \ldots, y_{ir-1}) \\ \ \wedge\ (y_{ir})_{i\in H} \in \mathcal{C}^{(r)}\big(H, (x_{ir})_{i\in H}\big) \\ \ \wedge\ \forall i \in H\ y_i = g_i(x_i, y_{i1}, \ldots, y_{i\ell}) \end{array} \right\}.$$

The goal of a protocol execution is to achieve consistency guarantees which are at least as strong as the guarantees of some target specification $\mathcal{C}$. This is the case, if the consistency guarantees of the protocol specification $\Pi\vec{\mathcal{C}}$ are at least as strong as the ones of the target specification $\mathcal{C}$. As already argued above, the consistency guarantee becomes stronger, as the set of possible outputs becomes smaller. We therefore say that a protocol $\Pi$ *constructs* a consistency specification $\mathcal{C}$ from the tuple $\vec{\mathcal{C}}$, if the set of possible outputs of the protocol specification $\Pi\vec{\mathcal{C}}(H,\vec{x}_H)$ for arbitrary inputs $H, \vec{x}_H$ is a subset of the corresponding set of possible outputs $\mathcal{C}(H,\vec{x}_H)$ of the target specification $\mathcal{C}$.

**Definition 18.** *A protocol* $\Pi$ constructs *a specification* $\mathcal{C}$ *from the tuple* $\vec{\mathcal{C}}$ *if we have for all* $H \subseteq \mathcal{P}$ *and all* $\vec{x}_H$ $\Pi\vec{\mathcal{C}}(H,\vec{x}_H) \subseteq \mathcal{C}(H,\vec{x}_H)$.

Often, one is interested in a broader notion of construction where specifications from a set $\mathfrak{C}$ may be invoked arbitrarily often during a protocol execution.

**Definition 19.** *A specification* $\mathcal{C}$ *can be constructed from a set of specifications* $\mathfrak{C}$*, denoted by* $\mathfrak{C} \longrightarrow \mathcal{C}$*, if there exists a tuple* $\vec{\mathcal{C}}$ *of specifications from* $\mathfrak{C}$ *(parallel composition allowed) which allows to construct* $\mathcal{C}$.

The above definition naturally extends to a construction notion among sets of consistency specifications: A set of consistency specifications $\mathfrak{C}'$ is constructible from $\mathfrak{C}$, denoted by $\mathfrak{C} \longrightarrow \mathfrak{C}'$ if all $\mathcal{C} \in \mathfrak{C}'$ can be constructed from $\mathfrak{C}$. We note that this notion of construction is transitive in the straight-forward sense.

**Lemma 13.** *Let $\mathfrak{C}_1, \mathfrak{C}_2, \mathfrak{C}_3$ be sets of consistency specifications for $\mathcal{P}$. Suppose $\mathfrak{C}_1 \longrightarrow \mathfrak{C}_2$ and $\mathfrak{C}_2 \longrightarrow \mathfrak{C}_3$. Then $\mathfrak{C}_1 \longrightarrow \mathfrak{C}_3$.*

*Proof.* For any $\mathcal{C}_3 \in \mathfrak{C}_3$ exists a tuple $\vec{\mathcal{C}_2}$ of specifications in $\mathfrak{C}_2$ such that $\vec{\mathcal{C}_2} \longrightarrow \mathcal{C}_3$. And for any $\mathcal{C} \in \vec{\mathcal{C}_2}$ exists a tuple $\vec{\mathcal{C}_1}$ such that $\vec{\mathcal{C}_1} \longrightarrow \mathcal{C}$. Now consider the protocol for $\vec{\mathcal{C}_2} \longrightarrow \mathcal{C}_3$. Each time the parties want to invoke a $\mathcal{C} \in \vec{\mathcal{C}_2}$, they could instead execute the protocol which constructs $\mathcal{C}$. This results in an overall protocol which constructs $\mathcal{C}_3$ from specifications in $\mathfrak{C}_1$. □

**Probabilistic Protocols.** In a probabilistic protocol, the parties may additionally use local randomness during the protocol execution. Formally, probabilistic protocols are modeled as distributions over deterministic protocols.

**Definition 20.** *A* probabilistic protocol *$\ell$-round $\mathbf{\Pi}$ for tuple $\vec{\mathcal{C}_{\mathbf{\Pi}}}$ with input domains $\mathcal{D}$ and output domains $\mathcal{R}$ is a random variable (for some distribution) over a set of deterministic protocols of at most $\ell$-rounds for tuple $\vec{\mathcal{C}_{\mathbf{\Pi}}}$ with input domains $\mathcal{D}$ and output domains $\mathcal{R}$.*

Note that our definition allows for protocols where parties have access to correlated randomness. We denote by $\mathbf{\Pi}\vec{\mathcal{C}_{\mathbf{\Pi}}}$ the random variable over the protocol specifications for $\mathbf{\Pi}$ and $\vec{\mathcal{C}_{\mathbf{\Pi}}}$. A protocol constructs a target specification $\mathcal{C}$ within $\epsilon$ if with probability strictly larger than $1-\epsilon$ $\mathbf{\Pi}\vec{\mathcal{C}_{\mathbf{\Pi}}}$ provides better consistency guarantees than $\mathcal{C}$.

**Definition 21.** *A probabilistic protocol $\mathbf{\Pi}$ for tuple $\vec{\mathcal{C}_{\mathbf{\Pi}}}$ constructs $\mathcal{C}$ within $\epsilon$ if*

$$\min_{H,\vec{x}_H} \mathsf{P}\big(\mathbf{\Pi}\vec{\mathcal{C}_{\mathbf{\Pi}}}(H,\vec{x}_H) \subseteq \mathcal{C}(H,\vec{x}_H)\big) > 1-\epsilon.$$

*A construction is called* perfect *if $\epsilon = 0$. A specification $\mathcal{C}$ can be constructed within $\epsilon$ from a set $\mathfrak{C}$, denoted by $\mathfrak{C} \xrightarrow{\epsilon} \mathcal{C}$, if there exists a tuple $\vec{\mathcal{C}_{\mathbf{\Pi}}}$ from $\mathfrak{C}$ which allows to construct $\mathcal{C}$ within $\epsilon$.*

Note that any deterministic construction is a perfect construction.

## 4.3 Impossibility Proofs

In this section, we consider a generalized version of so called "scenario"-proofs (see, e.g., [FLM85]). This proof technique, a special type of proof by contradiction, is normally used to prove that a specification, e.g., broadcast, cannot be constructed from authenticated channels within some $\epsilon$. Here, we extend "scenario"-proofs to the setting where parties are given additional setup. This means we want to prove statements of the form "There is no construction of a specification $\mathcal{C}$ from given specifications $\mathfrak{C}$ within $\epsilon$" where $\mathfrak{C}$ is (an arbitrary) set of specification which contains at least the complete network of authenticated channels.

The general proof strategy is as follows. Assume that we want to prove that $\mathcal{C}$ cannot be constructed from $\mathfrak{C}$ within $\frac{1}{k}$ where $\textsc{Auth} \in \mathfrak{C}$. The corresponding "scenario"-proof works as follows.

Towards a contradiction, assume that there exists a protocol $\mathbf{\Pi}$ which allows to construct $\mathcal{C}$ from $\mathfrak{C}$ within $\frac{1}{k}$. This implies that for each party $P_i$ and for each input $x_i$, there exists a corresponding (probabilistic) protocol system $\Pi_i^{x_i}$ which executes the protocol part of $P_i$ for input $x_i$[1]. The protocol system of a party has for any other party an interface where it expects to communicate with the other parties' protocol system. This models that parties are pair-wise connected via authenticated channels. If the parties are given additional specifications in $\mathfrak{C}$ (e.g., broadcast channels for some parties) during the protocol execution, this is modeled via a system $R$ that provides the functionality of this specification. In this case, the protocol systems have an additional interface where they expect to be connected to $R$.

Next, the assumed protocol systems are connected to form a specific *configuration* $S$. We consider the output (vector) of selected systems in $S$ which we denote by the random variable $\mathbf{Y}$. Our goal is to show that the properties of $\mathbf{Y}$ contradict the assumption that there exists a construction of $\mathcal{C}$ within $\frac{1}{k}$.

We use $k$ different *scenarios* to obtain conditions on the outcome of $\mathbf{Y}$. Each scenario describes $S$ as a protocol execution among three parties where exactly one of them is dishonest. With the exception of two systems (for the two honest parties) all parts of $S$ are considered to be the 'attack strategy' of the dishonest party. The initial assumption

[1]Such a system can be instantiated, for example, as an interactive Turing machine.

implies that the outputs of the two honest parties in this scenario must satisfy some consistency guarantee with probability strictly more than $1-\frac{1}{k}$. This directly translates into a condition on $\mathbf{Y}$. Namely, one finds a set $A_i$ such that $\mathsf{P}(\mathbf{Y} \in A_i) > 1-\frac{1}{k}$. The $k$ scenarios are chosen such that the intersection of all $A_i$'s is empty and therefore $\mathsf{P}(\mathbf{Y} \in \bigcap_{i=1}^k A_i) = 0$. In this case, the following lemma implies that for at least one $A_i$, it must hold that $\mathsf{P}(\mathbf{Y} \in A_i) \leq 1-\frac{1}{k}$, thus contradicting the fact that for all $i$, $\mathsf{P}(\mathbf{Y} \in A_i) > 1-\frac{1}{k}$ (as required by the assumption of a construction within $\epsilon = \frac{1}{k}$).

**Lemma 14.** *Let $A_1, \ldots, A_k$ be sets with non-empty union $A = \bigcup_{i=1}^k A_i$ and let $\mathbf{Y}$ be a random variable over some set $U \supseteq A$ such that $\mathsf{P}(\mathbf{Y} \in \bigcap_{i=1}^k A_i) = 0$. Then $\min_i \mathsf{P}(\mathbf{Y} \in A_i) \leq 1-\frac{1}{k}$.*

*Proof.* For convenience we denote for any set $B$ by $\mathsf{P}(B)$ the probability $\mathsf{P}(\mathbf{Y} \in B)$. We denote by $\overline{B}$ the complement of $B$ in $U$. Using elementary set operations and the union bound we get

$$\mathsf{P}(\bigcap_{i=1}^k A_i) = 1 - \mathsf{P}(\bigcup_{i=1}^k \overline{A_i}) \tag{4.2}$$

$$\geq 1 - \sum_{i=1}^k \mathsf{P}(\overline{A_i}) = 1 - \sum_{i=1}^k (1-\mathsf{P}(A_i)) = 1 - k + \sum_{i=1}^k \mathsf{P}(A_i). \tag{4.3}$$

As the minimum over all $\mathsf{P}(\mathbf{Y} \in A_i)$ is smaller than the average we finally get

$$\min_i \mathsf{P}(\mathbf{Y} \in A_i) \leq \frac{1}{k}\sum_{i=1}^k \mathsf{P}(A_i) \leq \frac{1}{k}\big(k-1+\mathsf{P}(\bigcap_{i=1}^k A_i)\big) = 1-\frac{1}{k}. \tag{4.4}$$

□

### 4.3.1 Broadcast Impossibility

In this section we prove as the-well known result from [KY84] that broadcast cannot be constructed from authenticated channels within $\frac{1}{3}$.

**Lemma 15.** *[KY84]* $\textsc{Auth} \overset{\frac{1}{3}}{\not\rightarrow} \mathrm{BC}_1$.

*Proof.* Towards a contradiction, let us assume that there exists a protocol $\mathbf{\Pi}$ which allows to construct $\mathrm{BC}_1$ from $\textsc{Auth}$ within $\frac{1}{3}$. Then there exist protocol systems $\Pi_1^0, \Pi_1^1, \Pi_2, \Pi_3$. Note that only the system of $P_1$ has an input. Each of these systems has two interfaces where it expects to be connected to the systems of the other two parties.

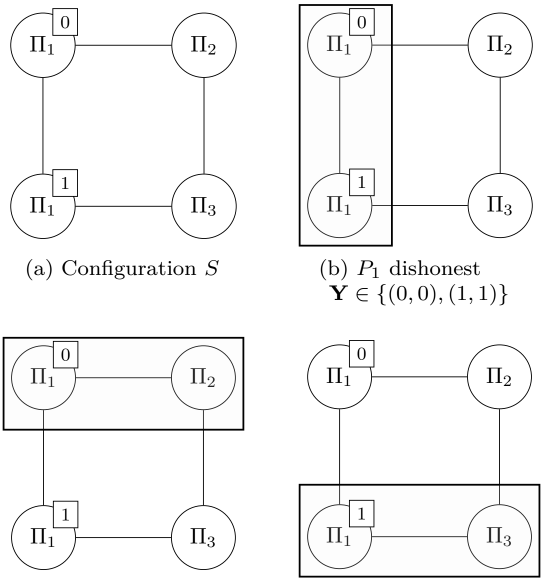
We consider the configuration $S$ in Figure 4.1a where all four systems are arranged in a circle. The random variable $\mathbf{Y}$ describes the output behavior of systems $\Pi_2$ and $\Pi_3$. This means that $\mathbf{Y}$ maps to bit-tuples where the first component represents the output of $\Pi_2$.

We examine the distribution of $\mathbf{Y}$ using different protocol execution scenarios. First, we consider the scenario where $P_2$ and $P_3$ are honest while $P_1$ is dishonest, i.e., $H = \{P_2, P_3\}$. In this scenario, consistency of broadcast ensures that the outputs of $P_2$ and $P_3$ are with probability strictly larger than $1 - \frac{1}{3}$ the same (independently of the behavior of $P_1$). In the configuration $S$, this corresponds to the scenario where the system of $P_1$ consists of the two left-most systems (cf. Figure 4.1b). This implies that $\mathbf{Y}$ is in $A_1 = \{(0,0),(1,1)\}$ with probability strictly larger than $1 - \frac{1}{3}$. Next, we consider the scenario where $P_1$ and $P_3$ are honest ($H = \{P_1, P_3\}$) and $P_1$ has input 1. In our configuration $S$, we can perceive the two systems on the top as the system of the dishonest $P_2$ (cf. Figure 4.1c). This implies (validity of broadcast) that $\mathsf{P}(\mathbf{Y} \in A_2) > 1 - \frac{1}{3}$ for $A_2 = \{(0,1),(1,1)\}$. Finally, we consider the case $H = \{P_1, P_3\}$ where $P_1$ has input 0. In our configuration $S$, we can perceive the two systems at the bottom as the system of the dishonest $P_3$ (cf. Figure 4.1d). This implies (validity of broadcast) that $\mathsf{P}(\mathbf{Y} \in A_3) > 1 - \frac{1}{3}$ for $A_3 = \{(0,0),(0,1)\}$.

We observe that $A_1 \cap A_2 \cap A_3 = \emptyset$ and thus $\mathsf{P}(\mathbf{Y} \in \bigcap_{i=1}^{3} A_i) = 0$. This implies with Lemma 14 that for at least one $A_i$, $\mathsf{P}(\mathbf{Y} \in A_i) \leq 1 - \frac{1}{3}$. This is a contradiction to the fact that $\mathsf{P}(\mathbf{Y} \in A_i) > 1 - \frac{1}{3}$ for all $A_i$, as required by the definition of a construction within $\epsilon = \frac{1}{3}$. Thus, there exists no $\epsilon$-construction of broadcast for $\epsilon \leq \frac{1}{3}$. □

### 4.3.2 Strong Broadcast Impossibility

In this section we prove a stronger impossibility for the construction of broadcast. That is, we show that broadcast channels, e.g. $\mathrm{BC}_1$, cannot



(a) Configuration $S$

(b) $P_1$ dishonest
$\mathbf{Y} \in \{(0,0),(1,1)\}$

(c) $P_2$ dishonest
$\mathbf{Y} \in \{(0,1),(1,1)\}$

(d) $P_3$ dishonest
$\mathbf{Y} \in \{(0,0),(0,1)\}$

Figure 4.1: The configuration $S$ and the three scenarios

be constructed within $\frac{1}{3}$ even if all other broadcast channels are available. Note that the result implies Lemma 15.

**Theorem 3.** $\{\textsc{Auth}, \mathrm{BC}_2, \mathrm{BC}_3\} \overset{\frac{1}{3}}{\not\to} \mathrm{BC}_1$.

*Proof.* To prove this result we use the “scenario”-proof technique introduced above. Assume therefore that there exists a probabilistic protocol $\mathbf{\Pi}$ which allows to construct $\mathrm{BC}_1$ from $\{\textsc{Auth}, \mathrm{BC}_2, \mathrm{BC}_3\}$ within $\epsilon = \frac{1}{3}$. Thus there exist protocol systems $\Pi_1^b, \Pi_2, \Pi_3$ where $b$ denotes the input bit of $P_1$. Additionally there exists a system $[\mathrm{BC}_2, \mathrm{BC}_3]$ which corresponds to the given broadcast channels for $P_2$ and $P_3$.

We first show how to construct a system $\overline{\mathrm{BC}}$ from system $[\mathrm{BC}_2, \mathrm{BC}_3]$. This system $\overline{\mathrm{BC}}$ will be used to build the configuration $S$, rather than $[\mathrm{BC}_2, \mathrm{BC}_3]$ directly. System $\overline{\mathrm{BC}}$ is essentially the same as $[\mathrm{BC}_2, \mathrm{BC}_3]$

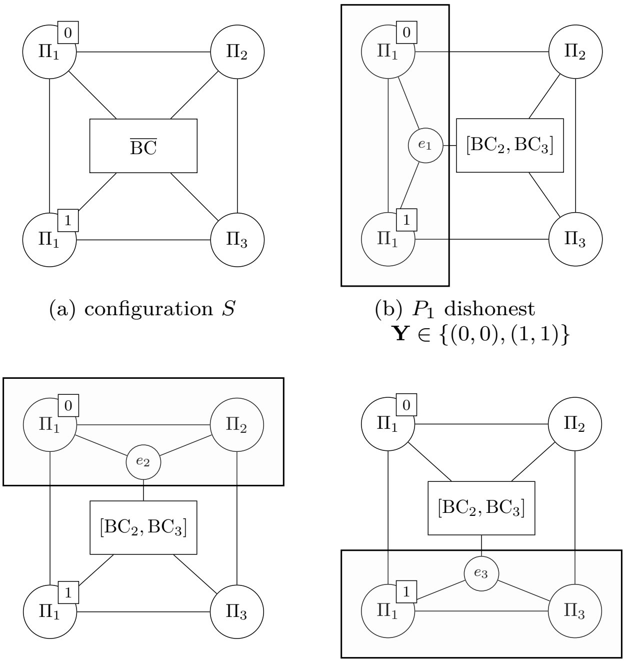
except that the interface of $P_1$ is cloned.

System $\overline{\mathrm{BC}}$ can be built from $[\mathrm{BC}_2, \mathrm{BC}_3]$ in three different ways. First, one can build it by adding a system $e_1$ to the $P_1$-interface of $[\mathrm{BC}_2, \mathrm{BC}_3]$ which relays the outputs of this interface to the two $P_1$-interfaces of $\overline{\mathrm{BC}}$. Second, one can build $\overline{\mathrm{BC}}$ from $[\mathrm{BC}_2, \mathrm{BC}_3]$ by adding a system $e_2$ to the $P_2$-interface of $[\mathrm{BC}_2, \mathrm{BC}_3]$. System $e_2$ relays any input at the $\overline{\mathrm{BC}}$ $P_2$-interface to $[\mathrm{BC}_2, \mathrm{BC}_3]$. Any output at the $P_2$-interface of $[\mathrm{BC}_2, \mathrm{BC}_3]$ is relayed to the $\overline{\mathrm{BC}}$ $P_1$-interface and the $\overline{\mathrm{BC}}$ $P_2$-interface of $e_2$. Analogously, one can build $\overline{\mathrm{BC}}$ from $[\mathrm{BC}_2, \mathrm{BC}_3]$ by adding a system $e_3$ to the $P_3$-interface of $[\mathrm{BC}_2, \mathrm{BC}_3]$. In summary we have that the systems $\overline{\mathrm{BC}}$, $e_1[\mathrm{BC}_2, \mathrm{BC}_3]$, $e_2[\mathrm{BC}_2, \mathrm{BC}_3]$, and $e_3[\mathrm{BC}_2, \mathrm{BC}_3]$ are all equivalent, i.e., the have exactly the same input/output behavior.

We consider now the configuration $S$ in Figure 4.2a where the four protocol systems $\Pi_1^0$, $\Pi_1^1$, $\Pi_3$, and $\Pi_2$ form a ring and are all connected to $\overline{\mathrm{BC}}$. We, denote by $\mathbf{Y}$ of systems $\Pi_2$ and $\Pi_3$. It follows from the above argumentation that systems $\Pi_2$ and $\Pi_3$ in the configurations in Figures 4.2b-4.2d must have the same output behavior $\mathbf{Y}$ as the systems $\Pi_2$ and $\Pi_3$ in configuration $S$.

We examine the distribution of $\mathbf{Y}$ using different protocol execution scenarios. First, we consider the scenario where $P_1$ is dishonest, i.e, $H = \{P_2, P_3\}$. The consistency of $\mathrm{BC}_1$ implies that with probability strictly larger $1 - \frac{1}{3}$ the outputs of $P_2$ and $P_3$ are the same. In this scenario $P_1$ could use a system consisting of the three left-most systems in Figure 4.2b. The consistency of broadcast thus implies for $S$ that $\mathsf{P}(\mathbf{Y} \in A_1) > 1 - \frac{1}{3}$ for $A_1 = \{(0,0),(1,1)\}$. Next, we consider the scenario $H = \{P_1, P_3\}$ where $P_1$ has input 1. Here, dishonest $P_2$ could run the top-three systems in Figure 4.2c. The validity of $\mathrm{BC}_1$ thus implies that $\mathsf{P}(\mathbf{Y} \in A_2) > 1 - \frac{1}{3}$ for $A_2 = \{(0,1),(1,1)\}$. Finally, we consider the scenario $H = \{P_1, P_2\}$ where $P_1$ has input 0. Here, dishonest $P_3$ could run the bottom-three systems in Figure 4.2d. The validity of $\mathrm{BC}_1$ thus implies that $\mathsf{P}(\mathbf{Y} \in A_3) > 1 - \frac{1}{3}$ for $A_3 = \{(0,0),(0,1)\}$. The intersection $A_1 \cap A_2 \cap A_3$ is empty and hence $\mathsf{P}(\mathbf{Y} \in A_1 \cap A_2 \cap A_3) = 0$. This implies with Lemma 14 that for at least one $A_i$, $\mathsf{P}(\mathbf{Y} \in A_i) \leq 1 - \frac{1}{3}$. This is a contradiction to the fact that $\mathsf{P}(\mathbf{Y} \in A_i) > 1 - \frac{1}{3}$ for all $A_i$ as required by the definition of a construction within $\epsilon = \frac{1}{3}$. This implies that there cannot exist a construction of broadcast $\mathrm{BC}_1$ from $\{\textsc{Auth}, \mathrm{BC}_2, \mathrm{BC}_3\}$ within $\epsilon = \frac{1}{3}$.

□



(a) configuration $S$

(b) $P_1$ dishonest
$\mathbf{Y} \in \{(0,0),(1,1)\}$

(c) $P_2$ dishonest
$\mathbf{Y} \in \{(0,1),(1,1)\}$

(d) $P_3$ dishonest
$\mathbf{Y} \in \{(0,0),(0,1)\}$

Figure 4.2: The configuration $S$ and the three scenarios.

## 4.4 Classification of Specifications

This section considers the classification of (sets of) consistency specifications according to their closures. The closure of a consistency specification set $\mathfrak{C}$ with respect to a consistency specification set $\mathfrak{T}$ contains all specifications in $\mathfrak{T}$ which can be constructed from $\mathfrak{C}$.

**Definition 22.** *The (relative)* closure *of* $\mathfrak{C}$ *with respect to* $\mathfrak{T}$ *is defined as* $\langle \mathfrak{C} \rangle_{\mathfrak{T}} := \{\mathcal{C} \in \mathfrak{T} \mid \mathfrak{C} \longrightarrow \mathcal{C}\}$.

The closure is monotone, i.e., for $\mathfrak{C}' \subseteq \mathfrak{C}$ it holds that $\langle \mathfrak{C}' \rangle_{\mathfrak{T}} \subseteq \langle \mathfrak{C} \rangle_{\mathfrak{T}}$. We will omit $\mathfrak{T}$, if clear from the context, and simply write $\langle \mathfrak{C} \rangle$. To classify a collection $\mathbf{S} = \{\mathfrak{C}_1, \ldots, \mathfrak{C}_k\}$ of consistency specification sets with respect to $\mathfrak{T}$ one can consider the different closures of the sets in $\mathbf{S}$.

**Definition 23.** *A* classification *of a collection* $\mathbf{S}$ *of consistency specification sets with respect to* $\mathfrak{T}$ *is the set* $\{\langle\mathfrak{C}\rangle_{\mathfrak{T}} \mid \mathfrak{C} \in \mathbf{S}\}$. *Two sets* $\mathfrak{C}, \mathfrak{C}' \in \mathbf{S}$ *realize the same class if* $\langle\mathfrak{C}\rangle_{\mathfrak{T}} = \langle\mathfrak{C}'\rangle_{\mathfrak{T}}$.

**Remark.** *For simplicity, we classify specifications according to perfect constructions, i.e., for constructions within* $\epsilon = 0$. *However, the classification presented in Section 4.4.1 would remain the same even if one allows for constructions within* $\epsilon \leq \frac{1}{3}$.

### 4.4.1 Classification of Single-Input Specifications

The goal of this section is to provide a motivating example of a consistency specifications classification. For this purpose we consider specifications for three parties $\mathcal{P} = \{P_1, P_2, P_3\}$ where $P_1$ has a binary input and the other two parties have binary outputs. We denote the set of all those specifications by $\mathfrak{T}_1$. For simplicity, we will omit the formal inputs (resp. outputs) of parties which have no input (resp. no output). For example, we will write $b \in \mathcal{C}(\{P_1, P_2\}, b)$ where $b$ is the input of $P_1$.

We denote by $\widehat{\mathrm{BC}}_1 \in \mathfrak{T}_1$ the broadcast channel for $P_1$ where (in contrast to $\mathrm{BC}_1$) party $P_1$ gets no output. More formally, we have

$$\widehat{\mathrm{BC}}_1(H, \vec{x}_H) = \left\{ \vec{y}_H \in \{0,1\}^H \;\middle|\; \begin{array}{l} \exists v\ \big( (\forall j \in H \setminus \{1\} : \vec{y}_{H|\{j\}} = v) \\ \wedge\ (1 \in H \Rightarrow v = \vec{x}_{H|\{1\}})\ \big) \end{array} \right\}.$$

for all $H \in \{P_1, P_2, P_3\}$ and all input vectors $\vec{x}_H$.

For the classification we assume that parties are pairwise connected by authenticated channels and have additionally access to a subset of specifications from $\mathfrak{T}_1$. Formally we thus consider a classification of the collection $\mathbf{S} := \{\mathfrak{C} \cup \textsc{Auth} \mid \mathfrak{C} \subseteq \mathfrak{T}_1\}$ with respect to $\mathfrak{T}_1$ where $\textsc{Auth}$ is the set of all authenticated channels for $\mathcal{P}$. We will show that $\mathfrak{T}_1$ is divided into two classes. First, we have the class $\langle\textsc{Auth}\rangle$ consisting of all specifications which can be constructed from authenticated channels. Second, we have the complement $\mathfrak{T}_1 \setminus \langle\textsc{Auth}\rangle$ which consists of all specifications which allow to construct broadcast $\widehat{\mathrm{BC}}_1$ given authenticated channels.

**Theorem 4.** *Given authenticated channels and a set of specifications $\mathfrak{C} \subseteq \mathfrak{T}_1$ one can either construct everything or just specifications which can be constructed from authenticated channels. In other words either $\langle \mathfrak{C} \cup \textsc{Auth} \rangle = \langle \{\widehat{\mathrm{BC}_1}\} \rangle = \mathfrak{T}_1$ or $\langle \mathfrak{C} \cup \textsc{Auth} \rangle = \langle \textsc{Auth} \rangle$.*

**Closure of Authenticated Channels.** A sufficient and necessary condition for $\mathcal{C} \in \langle \textsc{Auth} \rangle$ can be found by looking at the possible outputs of $\mathcal{C}$ in the case of two honest parties. To this end we define the following five sets of binary tuples.

$$\begin{aligned} M_{\mathcal{C}} &= \{(y_2, y_3) \mid (y_2, y_3) \in \mathcal{C}(\{P_2, P_3\})\} \\ M_{2,\mathcal{C}}^{(b)} &= \{(y_2, y_3) \mid y_2 \in \mathcal{C}(\{P_1, P_2\}, b)\} \quad \forall b \in \{0,1\} \\ M_{3,\mathcal{C}}^{(b)} &= \{(y_2, y_3) \mid y_3 \in \mathcal{C}(\{P_1, P_3\}, b)\} \quad \forall b \in \{0,1\} \end{aligned}$$

A specification $\mathcal{C}$ is in $\langle \textsc{Auth} \rangle$ if and only if

$$M_{2,\mathcal{C}}^{(0)} \cap M_{\mathcal{C}} \cap M_{3,\mathcal{C}}^{(1)} \neq \emptyset \quad \text{and} \quad M_{2,\mathcal{C}}^{(1)} \cap M_{\mathcal{C}} \cap M_{3,\mathcal{C}}^{(0)} \neq \emptyset. \tag{4.5}$$

We prove this condition in the following two lemmata.

**Lemma 16.** *A specification $\mathcal{C} \in \mathfrak{T}_1$ can be constructed from authenticated channels, i.e., $\mathcal{C} \in \langle \textsc{Auth} \rangle$, if condition (4.5) holds.*

*Proof.* To construct $\mathcal{C}$ from authenticated channels consider the following protocol $\Pi$. First, party $P_1$ sends its input bit to the other parties which exchange the received bits (cf. Figure 4.3). The output of $P_2$ is $g_2(b_2, b_{2,3})$ for a function $g_2 : \{0,1\}^2 \to \{0,1\}$ where $b_2$ and $b_{2,3}$ are the bits received from $P_1$ and $P_3$. Analogously, $P_3$ outputs $g_3(b_{3,2}, b_3)$. As we have

$$M_{2,\mathcal{C}}^{(0)} \cap M_{\mathcal{C}} \cap M_{3,\mathcal{C}}^{(1)} \neq \emptyset \quad \text{and} \quad M_{2,\mathcal{C}}^{(1)} \cap M_{\mathcal{C}} \cap M_{3,\mathcal{C}}^{(0)} \neq \emptyset.$$

we can define $g_2$ and $g_3$ as follows. For any bit $b \in \{0,1\}$ let

$$\big(g_2(b,b), g_3(b,b)\big) \in \mathcal{C}(\{P_1, P_2, P_3\}, b) \tag{4.6}$$

and

$$\big(g_2(b,1-b), g_3(b,1-b)\big) \in M_{2,\mathcal{C}}^{(b)} \cap M_{\mathcal{C}} \cap M_{3,\mathcal{C}}^{(1-b)}. \tag{4.7}$$

Note that for any $b \in \{0,1\}$ we have $\mathcal{C}(\{P_1, P_2, P_3\}, b) \subseteq M_{2,\mathcal{C}}^{(b)} \cap M_{\mathcal{C}} \cap M_{3,\mathcal{C}}^{(b)}$ and thus $M_{2,\mathcal{C}}^{(x)} \cap M_{\mathcal{C}} \cap M_{3,\mathcal{C}}^{(y)} \neq \emptyset$ for any $x, y \in \{0,1\}$. Consider now the

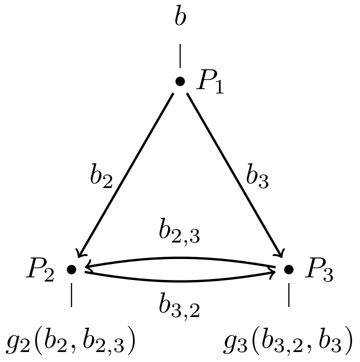


Figure 4.3: Protocol for Lemma 16.

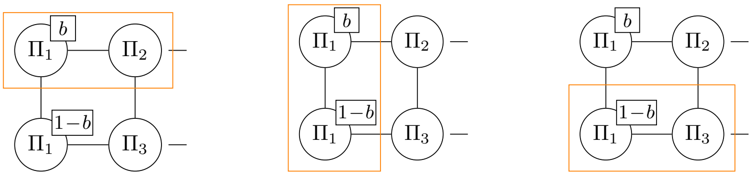
following cases. If everyone is honest, we have $b = b_2 = b_3 = b_{2,3} = b_{3,2}$. The output of $P_2$ and $P_3$ is thus $(g_2(b, b), g_3(b, b)) \in \mathcal{C}(\{P_1, P_2, P_3\}, b)$. For $H = \{P_1, P_2\}$ we have $b_2 = b$ and the output of $P_2$ is therefore $g_2(b, b_{2,3}) \in (M_{2,\mathcal{C}}^{(b)} \cap M_{\mathcal{C}} \cap M_{3,\mathcal{C}}^{(b_{2,3})})|_{P_2}$. Thus by the definition of $M_{2,\mathcal{C}}^{(b)}$ it holds that $g_2(b, b_{2,3}) \in \mathcal{C}(\{P_1, P_2\}, b)$. For $H = \{P_1, P_3\}$ it similarly holds that $b_3 = b$ and $g_3(b_{3,2}, b) \in \mathcal{C}(\{P_1, P_3\}, b)$. If $H = \{P_2, P_3\}$, it holds that $b_2 = b_{3,2}$ and $b_3 = b_{2,3}$. The output of the parties is therefore $(g_2(b_2, b_3), g_3(b_2, b_3)) \in M_{2,\mathcal{C}}^{(b_2)} \cap M_{\mathcal{C}} \cap M_{3,\mathcal{C}}^{(b_3)}$ and thus $(g_2(b_2, b_3), g_3(b_2, b_3)) \in M_{\mathcal{C}} = \mathcal{C}(\{P_2, P_3\})$. All the other cases follow directly from the monotonicity of $\mathcal{C}$. The protocol $\Pi$ therefore constructs $\mathcal{C}$ from authenticated channels. $\square$

Next, we show that Condition (4.5) is also necessary for $\mathcal{C} \in \langle \text{AUTH} \rangle$.

**Lemma 17.** *A specification $\mathcal{C} \in \mathfrak{T}_1$ with $M_{2,\mathcal{C}}^{(0)} \cap M_{\mathcal{C}} \cap M_{3,\mathcal{C}}^{(1)} = \emptyset$ or $M_{2,\mathcal{C}}^{(1)} \cap M_{\mathcal{C}} \cap M_{3,\mathcal{C}}^{(0)} = \emptyset$ cannot be constructed from authenticated channels (within $\frac{1}{3}$), i.e., $\mathcal{C} \notin \langle \text{AUTH} \rangle$.*

*Proof.* We use the proof technique from Section 4.3. Consider a $\mathcal{C} \in \mathfrak{T}_1$ with $M_{2,\mathcal{C}}^{(b)} \cap M_{\mathcal{C}} \cap M_{3,\mathcal{C}}^{(1-b)} = \emptyset$ for some $b \in \{0, 1\}$. Towards a contradiction, let us assume that there exists a protocol $\Pi$ which allows to construct $\mathcal{C}$ from AUTH within $\frac{1}{3}$. Then there exist systems $\Pi_1, \Pi_2, \Pi_3$ for parties

$P_1, P_2, P_3$. We consider the configuration in Figure 4.4 where all four systems are arranged in a circle. The random variable $\mathbf{Y}$ describes the output behavior of systems $\Pi_2$ and $\Pi_3$. First, consider $H = \{P_1, P_3\}$



(a) $P_2$ dishonest (b) $P_1$ dishonest (c) $P_3$ dishonest

Figure 4.4: Corruption Scenarios for Lemma 17.

where $P_1$ has input $1-b$ (cf. Figure 4.4a). As we assume that the protocol constructs $\mathcal{C}$, it must hold that $\mathbf{Y} \in M_{3,\mathcal{C}}^{(1-b)}$. Next, consider $H = \{P_2, P_3\}$ (cf. Figure 4.4b). Here, it must hold that $\mathbf{Y} \in M_{\mathcal{C}}$. Finally, consider $H = \{P_1, P_2\}$ where $P_1$ has input $b$ (cf. Figure 4.4c). Here, it must hold that $\mathbf{Y} \in M_{2,\mathcal{C}}^{(b)}$. We therefore have $\mathbf{Y} \in M_{2,\mathcal{C}}^{(b)} \cap M_{\mathcal{C}} \cap M_{3,\mathcal{C}}^{(1-b)}$ which is empty by the initial assumption on $\mathcal{C}$. This implies (cf. Lemma 15) that there is no construction of $\mathcal{C}$ from authenticated channels (within $\epsilon \leq \frac{1}{3}$). □

**Specifications Equivalent to Broadcast.** Observe that broadcast $\widehat{\mathrm{BC}}_1$ does not satisfy Condition (4.5), it can thus not be constructed from authenticated channels. Moreover, any specification in $\mathfrak{T}_1$ can be constructed from broadcast.

**Lemma 18.** *For any $\mathcal{C} \in \mathfrak{T}_1$, $\left\{\widehat{\mathrm{BC}}_1\right\} \longrightarrow \mathcal{C}$.*

*Proof.* The construction is trivially achieved with the following protocol. First, $P_1$ broadcasts its input $b$. Then parties $P_2, P_3$ output some fixed tuple in $\mathcal{C}(\{P_1, P_2, P_3\}, b)$. □

In the following we will show that any specification in $\mathcal{C} \in \mathfrak{T}_1 \setminus \langle \textsc{Auth} \rangle$ in addition to authenticated channels is enough to construct broadcast $\widehat{\mathrm{BC}}_1$. This implies that $\mathcal{C}$ (in addition of $\textsc{Auth}$) allows to construct any specification in $\mathfrak{T}_1$.

Recall that any specification $\mathcal{C} \in \mathfrak{T}_1 \setminus \langle \textsc{Auth} \rangle$ satisfies the condition

$$M_{2,\mathcal{C}}^{(0)} \cap M_{\mathcal{C}} \cap M_{3,\mathcal{C}}^{(1)} = \emptyset \quad \text{or} \quad M_{2,\mathcal{C}}^{(1)} \cap M_{\mathcal{C}} \cap M_{3,\mathcal{C}}^{(0)} = \emptyset \tag{4.8}$$

This condition implies that any $\mathcal{C} \in \mathfrak{T}_1 \setminus \langle \textsc{Auth} \rangle$ is equivalent[2] to a $\mathcal{C}' \in \mathfrak{T}_1 \setminus \langle \textsc{Auth} \rangle$ with $\mathcal{C}'(\{P_1, P_2, P_3\}, b) = \{(b, b)\}$ for $b \in \{0, 1\}$. We can therefore without loss of generality assume that $\mathcal{C}(\{P_1, P_2, P_3\}, b) = \{(b, b)\}$ for $b \in \{0, 1\}$. The condition also implies that

$$\begin{aligned}
&2 \leq |\mathcal{C}(\{P_2, P_3\})| \leq 3, \\
&\mathcal{C}(\{P_1, P_2\}, 0) \neq \mathcal{C}(\{P_1, P_2\}, 1), \text{ and} \\
&\mathcal{C}(\{P_1, P_3\}, 0) \neq \mathcal{C}(\{P_1, P_3\}, 1).
\end{aligned}$$

The specification $\mathcal{C}$ is thus similar to broadcast $\widehat{\mathrm{BC}}_1$ except that may offer weaker guarantees. More precisely, $\mathcal{C}$ can provide weaker validity for $P_2$, or weaker validity for $P_3$, or weaker consistency. We describe this weakening of $\mathcal{C}$ by a triple $(\alpha, \beta, \gamma)$ in $\{\Diamond, 0, 1\}^3$ where $\Diamond$ means that the specific component is not weakened at all.

**Definition 24.** *Let* $\alpha, \beta, \gamma \in \{\Diamond, 0, 1\}$. *The* weak cast $(\alpha, \beta, \gamma)$-$\mathrm{WC}_1$ *is defined as follows:*

- $(\alpha, \beta, \gamma)\text{-WC}_1(\{P_1, P_2, P_3\}, b) = \{(b, b)\}$
- $(\alpha, \beta, \gamma)\text{-WC}_1(\{P_1, P_2\}, b) = \begin{cases} \{b\} & \textit{if } b \neq \alpha \\ \{0, 1\} & \textit{if } b = \alpha \end{cases}$
- $(\alpha, \beta, \gamma)\text{-WC}_1(\{P_1, P_3\}, b) = \begin{cases} \{b\} & \textit{if } b \neq \beta \\ \{0, 1\} & \textit{if } b = \beta \end{cases}$
- $(\alpha, \beta, \Diamond)\text{-WC}_1(\{P_2, P_3\}) = \{(0, 0), (1, 1)\}$
  $(\alpha, \beta, 0)\text{-WC}_1(\{P_2, P_3\}) = \{(0, 0), (0, 1), (1, 1)\}$
  $(\alpha, \beta, 1)\text{-WC}_1(\{P_2, P_3\}) = \{(0, 0), (1, 0), (1, 1)\}$

Any specification in $\mathfrak{T}_1 \setminus \langle \textsc{Auth} \rangle$ is (equivalent) to a weak cast. For instance, $\widehat{\mathrm{BC}}_1 = (\Diamond, \Diamond, \Diamond)\text{-WC}_1$. Another example is $(\Diamond, \Diamond, 0)\text{-WC}_1$ which provides the validity condition of normal broadcast, but offers the weaker consistency guarantee $(\Diamond, \Diamond, 0)\text{-WC}_1(\{P_2, P_3\}) = \{(0, 0), (0, 1), (1, 1)\}$. We observe that not all weak cast are actually in $\mathfrak{T}_1 \setminus \langle \textsc{Auth} \rangle$.

[2]Two specifications $\mathcal{C}$ and $\mathcal{C}'$ are equivalent if $\{\mathcal{C}\} \longrightarrow \mathcal{C}'$ and $\{\mathcal{C}'\} \longrightarrow \mathcal{C}$.

**Lemma 19.** *For any $x \in \{\Diamond, 0, 1\}$ and $y \in \{0, 1\}$ the specifications $(x, y, y)$-$\mathrm{WC}_1$, $(y, x, 1-y)$-$\mathrm{WC}_1$, and $(y, y, x)$-$\mathrm{WC}_1$ are in $\langle\textsc{Auth}\rangle$.*

*Proof.* These specifications satisfy Condition (4.5). The statement follows with Lemma 16. $\square$

Next, we show that all other weak cast variants (in addition to $\textsc{Auth}$) allow to construct $\widehat{\mathrm{BC}}_1$. First, we consider weak casts were only the consistency guarantee is weakened, i.e. we look at $(\Diamond, \Diamond, \gamma)$-$\mathrm{WC}_1$ for $\gamma \in \{\Diamond, 0, 1\}$. Such a weak cast does not satisfy Condition (4.5) and is in $\mathfrak{T}_1 \setminus \langle\textsc{Auth}\rangle$ (cf. Lemma 17).

**Lemma 20.** *For any $\gamma \in \{\Diamond, 0, 1\}$,*

$$\textsc{Auth} \cup \{(\Diamond, \Diamond, \gamma)\text{-}\mathrm{WC}_1\} \longrightarrow \widehat{\mathrm{BC}}_1.$$

*Proof.* For $\gamma = \Diamond$ we have $\widehat{\mathrm{BC}}_1 = (\Diamond, \Diamond, \Diamond)$-$\mathrm{WC}_1$. For $\gamma \neq \Diamond$ and input bit $b$ of $P_1$ consider the following protocol. First, $P_1$ sends $(b, 1-b)$ to the other parties using two $(\Diamond, \Diamond, \gamma)$-$\mathrm{WC}_1$ invocations. Denote by $(b_2, c_2)$ (resp. $(b_3, c_3)$) the bits received by $P_2$ (resp. $P_3$). Then $P_2$ and $P_3$ exchange their bits using authenticated channels where $b_{2,3}, c_{2,3}$ (resp. $b_{3,2}, c_{3,2}$) denote the bits received by $P_2$ (resp. $P_3$). If $b_2 \neq c_2$, party $P_2$ outputs $b_2$. Otherwise, if $b_{2,3} \neq c_{2,3}$, $P_2$ outputs $b_{2,3}$. Otherwise $P_2$ outputs 0. The output of $P_3$ is computed analogous. Consider now the following cases. If everyone is honest, we have $b_2 = b_3 = b$ and $c_2 = c_3 = 1 - b$. The output of $P_2, P_3$ is thus $(b_2, b_3) = (b, b)$. For $H = \{P_1, P_2\}$ we have $b_2 = b$ and $c_2 = 1 - b$. The output of $P_2$ is therefore $b_2 = b$. For $H = \{P_1, P_3\}$ we have $b_3 = b$ and $c_3 = 1 - b$. The output of $P_3$ is therefore $b_3 = b$. If $H = \{P_2, P_3\}$, we have $(b_{2,3}, c_{2,3}) = (b_3, c_3)$ and $(b_{3,2}, c_{3,2}) = (b_2, c_2)$. If $b_2 \neq b_3$, we have $c_2 = c_3$ as $(1-\gamma, \gamma) \notin (\Diamond, \Diamond, \gamma)$-$\mathrm{WC}_1$. It is now easy to check that $P_2$ and $P_3$ will output the same bit. $\square$

Almost all weak-broadcast specifications where all three components are weakened are in $\langle\textsc{Auth}\rangle$ (cf. Lemma 19). The two exceptions are $(0, 1, 0)$-$\mathrm{WC}_1$ and $(1, 0, 1)$-$\mathrm{WC}_1$. Surprisingly one is able to construct broadcast from each of them given authenticated channels.

**Lemma 21.** *Both* $(0, 1, 0)$-$\mathrm{WC}_1$ *and* $(1, 0, 1)$-$\mathrm{WC}_1$ *(in addition to* $\textsc{Auth}$*) allow to construct* $\widehat{\mathrm{BC}}_1$.

*Proof.* With Lemma 20 it is enough to show that one can construct $(\Diamond,\Diamond,0)$-$\mathrm{WC}_1$ from $(0,1,0)$-$\mathrm{WC}_1$ (respectively that one can construct $(\Diamond,\Diamond,1)$-$\mathrm{WC}_1$ from $(1,0,1)$-$\mathrm{WC}_1$). We give the proof for the construction of $(\Diamond,\Diamond,0)$-$\mathrm{WC}_1$ from $(0,1,0)$-$\mathrm{WC}_1$. To this end, consider the following protocol.

First, $P_1$ sends its bit $b$ to the other parties using an authenticated channel, where $b_2$ (resp. $b_3$) denote the bit received by $P_2$ (resp. $P_3$). In the next step $P_1$ sends $b$ over $(0,1,0)$-$\mathrm{WC}_1$, where $c_2$ (resp. $c_3$) denotes the bit received by $P_2$ (resp. $P_3$). Finally, party $P_2$ outputs bit $o_2$ and party $P_3$ outputs bit $o_3$ where:

$$o_2 = \begin{cases} 1 & \text{if } (b_2,c_2)=(1,1) \\ 0 & \text{otherwise} \end{cases}$$

and

$$o_3 = \begin{cases} 0 & \text{if } (b_3,c_3)=(0,0) \\ 1 & \text{otherwise.} \end{cases}$$

Consider the following cases. If everyone is honest, we have $b_2 = c_2 = b_3 = c_3 = b$ the output of $P_2, P_3$ is thus $(o_2,o_3) = (b,b)$. For $H = \{P_1,P_2\}$ and $b = 0$ we have $(b_2,c_2) \neq (1,1)$ and thus $o_2 = 0$. If $b = 1$, we have $(b_2,c_2) = (1,1)$ and thus $o_2 = 1$. The case $H = \{P_1,P_3\}$ follows analogously to the case $H = \{P_1,P_2\}$. In the case $H = \{P_2,P_3\}$ the properties of $(0,1,0)-\mathrm{WC}_1$ imply that $(c_2,c_3) \neq (1,0)$. Thus it can not be that $(b_2,c_2) = (1,1)$ and $(b_3,c_3) = (0,0)$. The output $(o_2,o_3)$ is therefore in $(\Diamond,\Diamond,0)-\mathrm{WC}_1(\{P_2,P_3\})$. □

We observe that any other weak cast in $\mathfrak{T}_1 \setminus \langle \textsc{Auth} \rangle$ trivially allows to construct $(0,1,0)$-$\mathrm{WC}_1$ or $(1,0,1)$-$\mathrm{WC}_1$ as it offers stronger consistency guarantees. This implies together with Lemma 21 that one can construct $\widehat{\mathrm{BC}}_1$ from such a weak cast.

**Lemma 22.** *Let* $\mathcal{C} \in \mathfrak{T}_1 \setminus \langle \textsc{Auth} \rangle$*, then* $\textsc{Auth} \cup \{\mathcal{C}\} \rightarrow \mathrm{BC}_1$.

*Proof.* The consistency guarantees of $\mathcal{C}$ are strictly stronger than the ones of either $(0,1,0)$-$\mathrm{WC}_1$ or $(1,0,1)$-$\mathrm{WC}_1$. This allows to trivially construct at least one of them from $\mathcal{C}$. Lemma 20 and the transitivity of (deterministic) constructions imply that $\textsc{Auth} \cup \{\mathcal{C}\} \rightarrow \mathrm{BC}_1$ (cf. Lemma 21). □

We note that the availability of authenticated channels is crucial. For instance, one can show that $(\Diamond, \Diamond, 0)$-WC$_1$ is strictly weaker than BC$_1$, i.e., $\{(\Diamond, \Diamond, 0)\text{-WC}_1\} \not\to \text{BC}_1$.

## 4.5 Strong Separation Results

In this section we consider specifications for party set $\mathcal{P} = \{P_1, P_2, P_3\}$ where all inputs and outputs are bit-strings. The goal of this section is to prove a strong separation between broadcast and authenticated channels. That is, we present a specification, called XOR-cast, which neither can be constructed from authenticated channels within a constant $\epsilon$ nor is sufficient to construct broadcast within a constant $\epsilon$.

### 4.5.1 XOR-Cast

XOR-cast takes a bit $b_i$ from $P_i$ and a bit $b_j$ from $P_j$ as input. If all parties behave honestly, the value of $b_i \oplus b_j$ should be output by all parties. If one of the parties $P_i, P_j$ is dishonest, the honest parties should output the same value. If the third party $P_k$ is dishonest, the remaining parties should output $b_i \oplus b_j$.

**Definition 25.** *Let $P_i, P_j, P_k \in \mathcal{P}$ be pairwise distinct parties. The* XOR-cast *specification* XC$_{i,j}$ *for $P_i$ and $P_j$ is defined as*

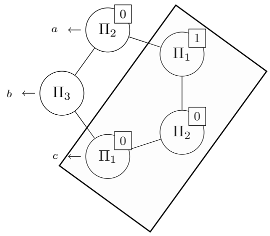
$$\text{XC}_{i,j}(H, \vec{x}_H) = \left\{ \vec{y} \in \{0,1\}^H \;\middle|\; \begin{array}{l} \exists v \; \big((\forall \ell \in H : \vec{y}_{H|\{\ell\}} = v) \\ \wedge \; (i, j \in H \Rightarrow v = \vec{x}_{H|\{i\}} \oplus \vec{x}_{H|\{j\}})\big) \end{array} \right\}.$$

*for all sets of honest parties $H$ and all honest inputs $\vec{x}_H$.*

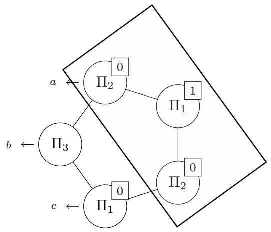
We first prove that XOR-cast cannot be constructed from the network of authenticated channels.

**Lemma 23.** *For all $i \neq j \in \{1, 2, 3\}$ $\{\text{Auth}\} \overset{\frac{1}{4}}{\not\to} \text{XC}_{i,j}$.*

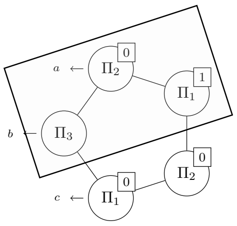
*Proof.* Without loss of generality, we show the statement for XC$_{1,2}$ using the "scenario"-proof technique from Section 4.3. Towards a contradiction, assume that there exists a protocol allowing to construct XC$_{1,2}$ from $\{\text{Auth}\}$ within $\frac{1}{4}$. Then there exist protocol systems $\Pi_1^{x_1}, \Pi_2^{x_2}, \Pi_3$ for parties $P_1, P_2, P_3$ where $x_1$ denotes the input bit of $P_1$ and $x_2$ denotes the input bit of $P_2$. Consider the pentagon configuration $S$ in Figure 4.5 and let $\mathbf{Y}$ be the random variable over the output $(a, b, c)$ of the three left-most systems.



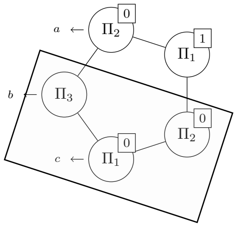
(a) $P_1$ dishonest
$\mathbf{Y} \in \{(0,0,0),(0,0,1),(1,1,0),(1,1,1)\}$



(b) $P_2$ dishonest
$\mathbf{Y} \in \{(0,0,0),(0,1,1),(1,0,0),(1,1,1)\}$



(c) $P_3$ dishonest, first strategy
$\mathbf{Y} \in \{(0,0,0),(0,1,0),(1,0,0),(1,1,0)\}$



(d) $P_3$ dishonest, second strategy
$\mathbf{Y} \in \{(1,0,0),(1,0,1),(1,1,0),(1,1,1)\}$

Figure 4.5: The configuration $S$ and the four scenarios.

We examine the distribution of $\mathbf{Y}$ using four different protocol execution scenarios. First, we consider the scenario where $P_2$ and $P_3$ are honest ($H = \{P_2, P_3\}$) and $P_2$ has input 0. In this scenario, the dishonest $P_1$ could run the three systems in the bottom-left in Figure 4.5a. The outputs of $P_2$ and $P_3$ must be the same. This implies $\mathsf{P}(\mathbf{Y} \in A_1) > 1 - \frac{1}{4}$ for $A_1 = \{(0,0,0),(0,0,1),(1,1,0),(1,1,1)\}$. Next, we consider the scenario $H = \{P_1, P_3\}$ where $P_1$ has input 0 (cf. Figure 4.5b). Here, the outputs of $P_1$ and $P_3$ must be the same. This implies that $\mathsf{P}(\mathbf{Y} \in A_2) > 1 - \frac{1}{4}$ for $A_2 = \{(0,0,0),(0,1,1),(1,0,0),(1,1,1)\}$. Next, we consider the scenario $H = \{P_1, P_2\}$ where both $P_1$ and $P_2$ have input 0 (cf. Figure 4.5c). Here, the output of $P_1$ must be $0 = 0 \oplus 0$. This implies that $\mathsf{P}(\mathbf{Y} \in A_3) > 1 - \frac{1}{4}$ for $A_3 = \{(0,0,0),(0,1,0),(1,0,0),(1,1,0)\}$. Finally, we consider the scenario $H = \{P_1, P_2\}$ where $P_1$ has input 1 and $P_2$ has input 0 (cf. Figure 4.5d). Here, the output of $P_2$ must be $1 = 1 \oplus 0$. This implies that $\mathsf{P}(\mathbf{Y} \in A_4) > 1 - \frac{1}{4}$ for $A_4 = \{(1,0,0),(1,0,1), (1,1,0),(1,1,1)\}$.

We observe that the intersection $A_1 \cap A_2 \cap A_3 \cap A_4$ is empty and hence $\mathsf{P}(\mathbf{Y} \in \bigcap_{i=1}^{4} A_i) = 0$. This implies with Lemma 14 that for at least one $A_i$, $\mathsf{P}(\mathbf{Y} \in A_i) \leq 1 - \frac{1}{4}$. This is a contradiction to the fact that $\mathsf{P}(\mathbf{Y} \in A_i) > 1 - \frac{1}{4}$ for all $A_i$ as required by the definition of a construction within $\epsilon = \frac{1}{4}$. Thus no construction of $\mathrm{XC}_{1,2}$ from Auth exists within $\frac{1}{4}$. □

Next, we show that one can perfectly construct $\mathrm{XC}_{i,j}$ given the complete network of authenticated channels and a broadcast channel for $P_i$ or $P_j$.

**Lemma 24.** *For all $i \neq j \in \{1,2,3\}$ $\{\text{Auth}, \mathrm{BC}_i\} \longrightarrow \mathrm{XC}_{i,j}$.*

*Proof.* Let $b_i$ be the input of $P_i$ and let $b_j$ be the input of $P_j$ and denote by $P_k$ the third party. Consider the following protocol.

1. $P_j$ sends $b_j$ to $P_i$. Denote by $\hat{b}_j$ the bit received by $P_i$.
2. $P_i$ broadcasts $b_k := b_i \oplus \hat{b}_j$ using $\mathrm{BC}_i$. Denote by $\hat{b}_k$ the bit received by $P_j$ and $P_k$.
3. $P_i$ outputs $b_k$, $P_j$ and $P_k$ both output $\hat{b}_k$.

If at least $P_i$ and $P_j$ are honest we have $\hat{b}_j = b_j$ and $\hat{b}_k = b_k$. All honest parties will output $b_k = b_i \oplus b_j$ as required by $\mathrm{XC}_{i,j}$. On the other hand if $H = \{P_j, P_k\}$ both honest parties will output $\hat{b}_k$ as required by $\mathrm{XC}_{i,j}$. If $H = \{P_i, P_k\}$ we have $\hat{b}_k = b_k$. Both honest parties will output $b_k$ as required by $\mathrm{XC}_{i,j}$. If at most one party is honest any output is fine, thus the protocol achieves the construction also in those cases. □

Finally, we show that XOR-cast is strictly weaker than broadcast. Even given all three XOR-casts, one cannot construct a single broadcast channel.

**Lemma 25.** *For all $i \in \{1,2,3\}$ $\{\mathrm{XC}_{1,2}, \mathrm{XC}_{1,3}, \mathrm{XC}_{2,3}, \text{Auth}\} \overset{\frac{1}{3}}{\not\longrightarrow} \mathrm{BC}_i$.*

*Proof.* Without loss of generality, we show that one cannot construct $\mathrm{BC}_1$ given all XOR-casts within $\epsilon \le \frac{1}{3}$. Towards a contradiction, let us assume that one can construct $\mathrm{BC}_1$ given the XOR-casts, i.e.,

$$\{\mathrm{XC}_{1,2}, \mathrm{XC}_{1,3}, \mathrm{XC}_{2,3}, \textsc{Auth}\} \longrightarrow \mathrm{BC}_1$$

within $\epsilon \le \frac{1}{3}$. Lemma 24 implies that one can perfectly construct all XOR-casts given broadcast channels $\mathrm{BC}_2, \mathrm{BC}_3$. This implies that one can construct $\mathrm{BC}_1$ from $\{\mathrm{BC}_2, \mathrm{BC}_3, \textsc{Auth}\}$ within $\epsilon \le \frac{1}{3}$, a contradiction to Lemma 3. $\square$ $\square$

The above lemmas directly imply the following theorem.

**Theorem 5.** *Authenticated channels and broadcast are strongly separated by XOR-cast.*

$$\begin{tikzcd}
\textsc{Auth} \arrow[r, "\frac{1}{4}", "/"'] & \mathrm{XC} \arrow[l, bend left, "\text{perf.}"] \arrow[r, "\frac{1}{3}", "/"'] & \mathrm{BC} \arrow[l, bend left, "\text{perf.}"]
\end{tikzcd}$$

## 4.5.2 Weak Broadcast

For comparison, we consider *weak broadcast* which was introduced in [Lam83]. This specification provides the same consistency guarantees as broadcast except that validity only holds if all parties are honest.

**Definition 26.** *Let $P_s \in \mathcal{P}$. A* weak broadcast-channel $\mathrm{wBC}_s$ *for sender $P_s$ is a $(\{0,1\}, \{0,1\})$- consistency specification where for every $H \subseteq \mathcal{P}$ and all $\vec{x}_H \in \{0,1\}^H$ it holds that*

$$\mathrm{wBC}_s(H, \vec{x}_H) = \left\{ \vec{y}_H \in \{0,1\}^H \;\middle|\; \begin{array}{l} \exists v \ \big( \ (\forall j \in H : \vec{y}_{H|\{j\}} = v) \\ \wedge \ (H = \mathcal{P} \Rightarrow v = \vec{x}_{H|\{s\}}) \ \big) \end{array} \right\}.$$

It was show in [Lam83] that weak broadcast cannot be constructed from authenticated channels using a deterministic protocol.

**Lemma 26.** *[Lam83] There exists no deterministic $r$-round protocol $\mathbf{\Pi}$ which allows for $\{\textsc{Auth}\} \longrightarrow \mathrm{wBC}_i$.*

*Proof.* Without loss of generality, let $P_1$ be the sender. Suppose there exists a deterministic $r$-round protocol $\mathbf{\Pi}$ which allows to construct $\text{wBC}_1$ from $\textsc{Auth}$. Then, there exist protocol systems $\mathbf{\Pi}_1^x, \mathbf{\Pi}_2, \mathbf{\Pi}_3$ for parties $P_1, P_2, P_3$, where $x$ denotes the input of $P_1$. Choose $k > r+1$ as a multiple of 3 and arrange $4k$ such systems in a ring as follows: Start with a system $\mathbf{\Pi}_1^0$ and continue with systems $\mathbf{\Pi}_2, \mathbf{\Pi}_3$; each system is connected via authenticated channels to its predecessor and successor. Now repeat this pattern going clockwise, until $2k$ systems have been connected in this manner. Because $k$ is a multiple of three, the last system in this arrangement will be a system $\mathbf{\Pi}_3$. Now, restart the pattern from the end of this arrangement, but instead of $\mathbf{\Pi}_1^0$, use $\mathbf{\Pi}_1^1$. Arrange another $2k$ nodes in this manner, thereby closing the ring.

Consider the system $\mathbf{\Pi}_1^0$ at "the top" of the ring. As all systems in the ring are deterministic the view of $\mathbf{\Pi}_1^0$ after $r$ rounds is the same as if the system were run in a triangular configuration (where the triangle consists of $\mathbf{\Pi}_1^0, \mathbf{\Pi}_2, \mathbf{\Pi}_3$). The validity of weak broadcast implies that the system $\mathbf{\Pi}_1^0$ must output 0. Similarly, the system $\mathbf{\Pi}_1^1$ at "the bottom" of the ring must output 1. Now, consider any to adjacent systems in the ring. One can view the rest of the ring as an attack strategy of a corrupted party. Thus by consistency of weak broadcast any two adjacent systems must output the same value. We thus arrive at a contradiction. □

On the other hand, the results of [FGH+02] imply that weak broadcast can be achieved from authenticated channels for any $\epsilon > 0$.

**Lemma 27.** *[FGH+02] For any* $\epsilon > 0$ $\{\textsc{Auth}\} \xrightarrow{\epsilon} \text{wBC}_i$.

Finally, we show that weak broadcast is separated from broadcast. More precisely, we show that broadcast allows to construct weak broadcast while on the other hand broadcast cannot be constructed from weak broadcast within $\epsilon \leq \frac{1}{3}$.

**Theorem 6.** *Weak broadcast and broadcast are strongly separated.*

The theorem follows from the following two lemmata.

**Lemma 28.** *For all* $i \in \{1,2,3\}$ $\{\text{BC}_i\} \longrightarrow \text{wBC}_i$.

*Proof.* For all $H$ and all $\vec{x}_H$ it holds that $\text{BC}_i(H, \vec{x}_H) \subseteq \text{wBC}_i(H, \vec{x}_H)$. This directly implies $\{\text{BC}_i\} \longrightarrow \text{wBC}_i$. □

**Lemma 29.** *For all $i \in \{1,2,3\}$ $\{\text{wBC}_i, \textsc{Auth}\} \overset{\frac{1}{3}}{\not\rightarrow} \text{BC}_i$.*

*Proof.* We first show that $\text{XC}_{i,j}$ for $j \neq i$ is enough to construct $\text{wBC}_i$. The following protocol $\pi$ allows $P_i$ to weak broadcast its bit $b$ using $\text{XC}_{i,j}$.

1. $\text{XC}_{i,j}$ is invoked where $P_i$ inputs $b$ and $P_j$ inputs 0. Denote by $b_i, b_j, b_k$ the bits the parties $P_i, P_j, P_k$ receive as output from $\text{XC}_{i,j}$.

2. $P_i$ outputs $b_i$, $P_j$ outputs $b_j$ and $P_k$ outputs $b_k$.

The properties of $\text{XC}_{i,j}$ ensure that honest parties will always output the same bit, as required by the consistency of $\text{wBC}_i$. If at least $P_i$ and $P_j$ are honest, the output of $\text{XC}_{i,j}$ is $b = b \oplus 0$. The protocol thus achieves the validity condition required by $\text{wBC}_i$.

From Lemma 25 we know that $\{\text{XC}_{i,j}, \textsc{Auth}\} \overset{\frac{1}{3}}{\not\rightarrow} \text{BC}_i$. This implies that $\text{BC}_i$ cannot be constructed from $\{\text{wBC}_i, \textsc{Auth}\}$ within $\epsilon \leq \frac{1}{3}$. □

## 4.6 Discussion and Open Problems

In this chapter we have proposed a classification of consistency specifications according to the consistency guarantees they allow to achieve. As a motivating example we have given a complete classification of specifications where a single party can give a binary input. This classification showed that such a single-input specification can either be constructed from authenticated channels or is enough to construct broadcast. We then considered three-party specifications where more than just one party may have an input. Here, we showed that XOR-cast strongly separates authenticated channels and broadcast. It is an open problem to find a complete classification of three-party specifications where parties have binary inputs and outputs.

# Chapter 5

# Efficient General-Adversary Multi-Party Computation

The content of this chapter is based on [HT13].

## 5.1 Introduction

In this chapter, we consider MPC in the presence of an active general-adversary. General adversaries are characterized by an adversary structure $\mathcal{Z}$ which enumerates all possible set of corrupted parties. Compared to threshold adversaries, general adversaries are more flexible, which is relevant in particular when the set of parties is not very large.

On the other hand, protocols secure against general adversaries are typically by orders of magnitude less efficient than protocols for threshold adversaries. Most threshold protocols communicate poly($n$) bits per multiplication, where general-adversary protocols require $|\mathcal{Z}|$ bits which is typically exponential in $n$. However, in some situations threshold protocols can not provide the necessary flexibility and one must one general adversary protocols. In the design of general adversary protocols the concrete communication complexity is therefore highly relevant. For

| Setting | Cond. | Bits / Mult. | Reference |
|---|---|---|---|
| passive perfect | $\mathcal{Q}^2$ | $\lvert\mathcal{Z}\rvert \cdot \text{poly}(n)$ | [Mau03] |
| active perfect | $\mathcal{Q}^3$ | $\lvert\mathcal{Z}\rvert^3 \cdot \text{poly}(n)$ | [Mau03] |
| active perfect | $\mathcal{Q}^3$ | $\lvert\mathcal{Z}\rvert^2 \cdot \text{poly}(n)$ | our result |
| active unconditional | $\mathcal{Q}^2$ | $\lvert\mathcal{Z}\rvert^3 \cdot \text{poly}(n, \kappa)$ | [Mau03]/[HMZ08] |
| active unconditional | $\mathcal{Q}^3$ | $\lvert\mathcal{Z}\rvert^2 \cdot \text{poly}(n, \kappa)$ | [PSR03] |
| active unconditional | $\mathcal{Q}^2$ | $\lvert\mathcal{Z}\rvert \cdot \text{poly}(n, \kappa)$ | our result |

Table 5.1: Communication complexity of different protocols

example for $n = 25$, $|\mathcal{Z}|$ is expected to be around one million, and a protocol communicating $|\mathcal{Z}| \cdot \text{poly}(n)$ might be acceptable, whereas a protocol communicating $|\mathcal{Z}|^3 \cdot \text{poly}(n)$ might be useless.

### 5.1.1 Contributions

In the statistically-secure model, one can tolerate at most adversary structures satisfying $\mathcal{Q}^2(\mathcal{P}, \mathcal{Z})$. The most efficient protocol known to date, which is also optimal in terms of resilience, requires $|\mathcal{Z}|^3 \cdot \text{poly}(n, \kappa)$ bits of communication (where $\kappa$ is the security parameter) [Mau03, HMZ08]. There exists a protocol with communication complexity of $|\mathcal{Z}|^2 \cdot \text{poly}(n, \kappa)$ [PSR03]. But this results is non-optimal in terms of resilience, as it tolerates only adversaries satisfying $\mathcal{Q}^3$.

Using a new approach for multiplication, we construct a protocol communicating $|\mathcal{Z}| \cdot \text{poly}(n, \kappa)$ bits and tolerating $\mathcal{Q}^2$ adversary structures. This protocol is optimal both in terms of overall efficiency and resilience. We stress that even with cryptographic security, $\mathcal{Q}^2$ is necessary and complexity linear in $|\mathcal{Z}|$ is required at least with respect to overall complexity of computation and communication (see [Hir01]).

Furthermore, we present a perfectly secure protocol (with no error probability) with communication complexity of $|\mathcal{Z}|^2 \cdot \text{poly}(n)$. It is optimal in terms of resilience ($\mathcal{Q}^3$) and also the most efficient protocol up to date in the model with perfect security.

We refer to Table 5.1 for an overview of the communication complexity of our protocols.

## 5.2 Preliminaries

We refer to Section 1.1 for a short introduction to secure multi-party computation in general. This section provides additional preliminaries for the case of general adversaries.

**Parties and Computation.** In this chapter we consider a set $\mathcal{P}$ of $n$ parties which want to compute a function $f$ over some finite field $\mathbb{F}$. The function is specified by a circuit $\mathcal{C}$ consisting of input, output, random, addition, and multiplication gates.

The parties are connected by a complete network of secure channels and have access to authenticated broadcast channels. We note that the broadcast channels can be emulated by the parties (see e.g. [FM98] or [PW96]).

**Adversary and Adversary Structure.** We consider an unbounded static active adversary $\mathcal{A}$. The corruption choice is limited by means of an adversary structure $\mathcal{Z} = \{Z_1, \ldots, Z_\ell\} \subseteq 2^{\mathcal{P}}$, i.e. all corrupted parties must be part of an adversary set in $\mathcal{Z}$. We denote the chosen set by $Z^*$. Note that $Z^*$ is not known to the honest parties and is solely used for ease of notation. We say that $\mathcal{Z}$ satisfies the $\mathcal{Q}^k(\mathcal{P}, \mathcal{Z})$ property if $\forall Z_1, \ldots, Z_k \in \mathcal{Z} \quad \mathcal{P} \not\subseteq Z_1 \cup \cdots \cup Z_k$.

**Security.** A protocol is $\mathcal{Z}$-secure if anything the adversary achieves during the execution of the protocol can be achieved in the ideal world as well. More precisely, for every adversary in the real world there exists an adversary in the ideal world such that both the information the adversary gets and the output of honest parties are statistically indistinguishable for perfect security respectively statistically close for unconditional security.

The main result from [HM97] states that $\mathcal{Q}^3(\mathcal{P}, \mathcal{Z})$ resp. $\mathcal{Q}^2(\mathcal{P}, \mathcal{Z})$ are the necessary and sufficient conditions for the existence of perfectly resp. unconditionally $\mathcal{Z}$-secure protocols considering active adversaries.

For simplicity we assume that all messages sent during the execution of $\Pi$ are from the right domain. If a party receives a message where this is not the case, he replaces it with an arbitrary element from the right domain. If a party receives an unexpected message, he ignores it.

## 5.3 Perfect Protocol

In this section we present a perfectly $\mathcal{Z}$-secure protocol for an arbitrary adversary structure $\mathcal{Z}$ satisfying the $\mathcal{Q}^3$ property. The communication complexity of the protocol is quadratic in $|\mathcal{Z}|$. The efficiency gain is due to an improved multiplication protocol. The sharing is (up to presentation) the same as in [Mau03].

### 5.3.1 Secret Sharing

Secret sharing allows a party to distribute a secret value among the party set, such that only qualified subsets of parties are able to reconstruct it. The secret sharing used for our protocol is based on the one from [Mau03] / [BFH$^+$08]. It is characterized by a *sharing specification* $\mathbb{S} = (S_1, \ldots, S_h)$, which is a tuple of subsets of $\mathcal{P}$.

**Definition 27.** *A value $s$ is* shared *with respect to sharing specification $\mathbb{S} = (S_1, \ldots, S_h)$ if the following holds:*

a*) There exist shares $s_1, \ldots, s_h$ such that $s = \sum_{q=1}^{h} s_q$*

b*) Each $s_q$ is known to every (honest) parties in $S_q$*

We denote the sharing of a value $s$ by $[s]$ and use $[s]_q$ as notation for $s_q$, the $q$-th share. A sharing specification $\mathbb{S} = (S_1, \ldots, S_h)$ is called *$\mathcal{Z}$-private* if for every $Z \in \mathcal{Z}$ there is an $S \in \mathbb{S}$ such that $Z \cap S = \emptyset$. A sharing specification $\mathbb{S} = (S_1, \ldots, S_h)$ and an adversary structure $\mathcal{Z}$ satisfy $\mathcal{Q}^k(\mathbb{S}, \mathcal{Z})$ if $S \not\subseteq Z_1 \cup \cdots \cup Z_k \quad \forall Z_1, \ldots, Z_k \in \mathcal{Z}\ S \in \mathbb{S}$. If $\mathbb{S}$ is $\mathcal{Z}$-private, a sharing $[s]$ does not leak information to the adversary, as all shares known by the adversary are statistically independent of $s$. The parties can compute a sharing of any linear combination of shared values (with respect to a sharing specification $\mathbb{S}$) by locally computing the linear combination of their shares. This property is called the linearity of the sharing. The following protocol Share allows a dealer $P_D$ to correctly share value $s$ among the parties in $\mathcal{P}$.

**Protocol Share**$(\mathcal{P}, \mathcal{Z}, \mathbb{S}, P_D, s)$ **[Mau03]**

```
0: The dealer $P_D$ takes $s$ as input.
1: $P_D$ splits $s$ into random shares $s_1, \dots, s_{|\mathbb{S}|}$ subject to $s = \sum_{q=1}^{|\mathbb{S}|} s_q$.
2: for all $q \in \{1, \dots, |\mathbb{S}|\}$ do
3:     $P_D$ sends $s_q$ to every party in $S_q$.
4:     Each party in $S_q$ forwards the received value to each party in $S_q$.
5:     Each party in $S_q$ checks that the received values are all the same
       and broadcasts OK, or NOK accordingly.
6:     If a party in $S_q$ broadcast NOK, the dealer broadcasts $s_q$ and
       the parties in $S_q$ take this value (resp. some default value if the
       dealer does not broadcast) as share. Otherwise every party in $S_q$
       takes the value it received in Step 3 as share.
7: end for
8: The parties in $\mathcal{P}$ collectively output $[s]$.
```

**Lemma 30.** *For any adversary structure $\mathcal{Z}$ protocol Share$(\mathcal{P}, \mathcal{Z}, \mathbb{S}, P_D, s)$ securely computes a sharing $[s']$. For honest $P_D$ it holds that $s' = s$. The protocol communicates at most $|\mathbb{S}|(n^2 + n) \log |\mathbb{F}|$ bits and broadcasts at most $|\mathbb{S}|(\log |\mathbb{F}| + n)$ bits.*

*Proof. Correctness:* For each $s_q$ either all the honest parties in $S_q$ hold the same value after Step 3, or one of them complains and they receive a consistent value in Step 6. Hence the protocol outputs a (consistent) sharing $[s']$. If the dealer is honest he is able to ensure in Steps 3 and 6 that the honest parties use the intended value for $s_q$ such that $s = s'$.

*Privacy:* Let the dealer be honest, as otherwise secrecy is trivially fulfilled. All a party learns beyond his designated output are values broadcast in Step 6. However this does not violate secrecy as these values are already known to the adversary (from Step 3).

*Complexity:* For each share at most $n + n^2$ values are sent and at most $n + \log |\mathbb{F}|$ bits broadcast. □

For publicly known value $s$ the parties can invoke DefaultShare to get a sharing $[s]$ without having to communicate.

**Protocol DefaultShare**$(\mathcal{P}, \mathcal{Z}, \mathbb{S}, s)$

```
0: Every party takes $s$ as input.
```

```
1: The share $s_1$ is set to $s$ and all other shares are set to 0.
2: The parties in $\mathcal{P}$ collectively output $[s]$.
```

**Lemma 31.** *DefaultShare$(\mathcal{P}, \mathcal{Z}, \mathbb{S}, s)$ securely computes a sharing $[s]$ where $s$ is a publicly known value. The protocol does not communicate.*

*Proof. Correctness:* In Step 1 every honest party in $S_q$ takes the same value for share $s_q$. As the sum of all shares is $s$, the protocol outputs a consistent sharing $[s]$.

*Privacy:* During the protocol no communication occurs, hence the adversary does not obtain new information. □

The protocol ReconstructShare allows the reconstruction of a share $[s]_q$ to the parties in some set $R$. This implies that the parties can reconstruct a shared value $[s]$ by invoking ReconstructShare for each share.

**Protocol ReconstructShare**$(\mathcal{P}, \mathcal{Z}, \mathbb{S}, [s]_q, R)$

```
0: The parties in $S_q$ take the share $[s]_q$ as input.
1: Every party $P_i$ in $S_q$ sends $[s]_q$ to every party in $R$.
2: For each party $P_j \in R$ let $v_{j,i}$ be the value received from $P_i$. Then
   $P_j$ outputs some value $v_j$ such that there exists a $Z \in \mathcal{Z}$ with
   $v_{j,i} = v_j$ for all $P_i \in S_q \setminus Z$.
```

**Lemma 32.** *If $S_q$ and $\mathcal{Z}$ satisfy $Q^2(S_q, \mathcal{Z})$, the protocol ReconstructShare securely reconstructs the share $[s]_q$ to the parties in $R$, such that every (honest) party outputs $[s]_q$. The protocol communicates at most $n^2 \log |\mathbb{F}|$ bits.*

*Proof. Correctness:* In Step 1 all honest parties will send the same value $[s]_q$, which is a suitable choice for $v_j$ for an (honest) party $P_j \in R$ in Step 2. For the sake of contradiction suppose there exist two values $v_1 \neq v_2$ with corresponding $Z_1, Z_2 \in \mathcal{Z}$ such that the condition of Step 2 holds for both of them. Hence $(S_q \setminus Z_1) \cap (S_q \setminus Z_2) = \emptyset$ and thus $S_q \subseteq Z_1 \cup Z_2$ which contradicts $Q^2(S_q, \mathcal{Z})$. Therefore every honest party outputs the value $[s]_q$.

*Privacy:* The adversary learns at most $[s]_q$ (if a malicious party is part of $R$).

*Complexity:* Each party in $S_q$ sends his value to at most $n$ parties. □

**Protocol Reconstruct**$(\mathcal{P}, \mathcal{Z}, \mathbb{S}, [s], R)$ **[Mau03]**

```
0: The parties in $\mathcal{P}$ take collectively $[s]$ as input.
1: $\forall q \in \{1, \ldots, |\mathbb{S}|\}$ protocol ReconstructShare$(\mathcal{P}, \mathcal{Z}, \mathbb{S}, [s]_q, R)$ is invoked.
2: The parties in $R$ locally sum up the obtained shares and output the sum $s$.
```

**Lemma 33.** *If $\mathbb{S}$ and $\mathcal{Z}$ satisfy $\mathcal{Q}^2(\mathbb{S}, \mathcal{Z})$ and $[s]$ is a sharing of the value $s$, then* Reconstruct$(\mathcal{P}, \mathcal{Z}, \mathbb{S}, [s], R)$ *securely reconstructs $s$ to the parties in $R$. The protocol communicates at most $|\mathbb{S}|n^2 \log |\mathbb{F}|$ bits.*

*Proof.* Correctness and privacy follow directly from Lemma 32. As ReconstructShare is invoked $|\mathbb{S}|$ times the complexity follows as well. □

## 5.3.2 Multiplication

We present a protocol for the perfectly-secure computation of the (shared) product of two shared values $[a]$ and $[b]$ (with respect to a sharing specification $\mathbb{S}$). Along the lines of [Mau03] the fundamental idea of multiplication is to assign each local product $a_p b_q$ to a party in $S_p \cap S_q$, who computes and shares his designated products. The sum of all these sharings is a sharing of $ab$ as long as no party actively cheated. So each party is mapped to a collection of local products, formalized by a function $I : [n] \to 2^{\{(p,q) \ | \ 1 \le p,q \le |\mathcal{Z}|\}}$ with the constraint that $\forall (p,q) \ \exists! \ i$ such that $(p,q) \in I(i)$. W.l.o.g let $I(i) := \{(p,q) \mid P_i = \min_P\{P \in S_p \cap S_q\}\}$.

We first show an optimistic multiplication protocol which takes an additional parameter $Z$ and computes the correct product if the actual adversary set $Z^*$ is a subset of $Z$. In this protocol local products are assigned to parties in $\mathcal{P} \setminus Z$ only. Clearly this is possible if and only if for each local product a party in $\mathcal{P} \setminus Z$ holds both involved shares, i.e. $\forall S_p, S_q \in \mathbb{S} \ : \ S_p \cap S_q \setminus Z \neq \emptyset$. So for each $Z \in \mathcal{Z}$ let $I_Z$ be a mapping as above with the additional constraint that $\forall P_i \in Z \ I_Z(i) = \emptyset$. Without loss of generality, let $I_Z(i) := \{(p,q) \mid P_i = \min_P\{P \in S_p \cap S_q \setminus Z\}\}$.

**Protocol OptimisticMult**$(\mathcal{P}, \mathcal{Z}, \mathbb{S}, [a], [b], Z)$

```
0: The parties in $\mathcal{P}$ take collectively $[a], [b]$ and $Z$ as input.
```

```
1:
    a) Each party $P_i \in \mathcal{P} \setminus Z$ (locally) computes his designated
       products and shares the sum $c_i = \sum_{(p,q) \in I_Z(i)} a_p b_q$.
    b) For each $P_i \in Z$ DefaultShare$(\mathcal{P}, \mathcal{Z}, \mathbb{S}, 0)$ is invoked to share
       $c_i = 0$.
2: The parties collectively output $([c_1], \ldots, [c_n])$ and $[c] = \sum_{i=1}^{n} [c_i]$.
```

**Lemma 34.** *Let $Z \subseteq \mathcal{P}$ such that $\forall S_p, S_q \in \mathbb{S} \,:\, S_p \cap S_q \setminus Z \neq \emptyset$. Then the protocol* OptimisticMult *securely computes sharings $[c], ([c_1], \ldots, [c_n])$. If no party in $\mathcal{P} \setminus Z$ actively cheats (in particular, if $Z^* \subseteq Z$), then $\forall i \; c_i = \sum_{(p,q) \in I_Z(i)} a_p b_q$ and $c = ab$. The protocol communicates at most $\mathcal{O}(|\mathbb{S}| n^3 \log |\mathbb{F}|)$ bits and broadcasts at most $\mathcal{O}(|\mathbb{S}|(n \log |\mathbb{F}| + n^2))$ bits.*

*Proof. Correctness:* The properties of the sharing protocol guarantee that the outputs are valid sharings. If none of the parties in $\mathcal{P} \setminus Z$ cheated actively, it holds for each $P_i$ that $c_i = \sum_{(p,q) \in I_Z(i)} a_p b_q$. The condition $\forall S_p, S_q \in \mathbb{S} \,:\, S_p \cap S_q \setminus Z \neq \emptyset$ guarantees that $ab = \sum_{i=1}^{n} \sum_{(p,q) \in I_Z(i)} a_p b_q$. Hence it follows that $c = ab$.

*Privacy and Complexity:* Follow directly from Lemmas 30 and 31. □

As the parties do not know the actual adversary set $Z^*$, they invoke OptimisticMult once for each set $Z \in \mathcal{Z}$ (Step 1 of the Multiplication protocol). This guarantees that at least one of the resulting sharings is correct.

By comparing them the parties can determine whether cheating occurred (Step 2 of the Multiplication protocol). If all sharings are equal, no cheating occurred and any of the sharings can serve as sharing of the product. Otherwise at least one party cheated. In this case the (honest) parties can identify him and remove all sharings where he was involved in computation, as these sharings are potentially tampered (Step 3 of the Multiplication protocol).

This checking and removing is repeated until all remaining sharing are equal (and hence correct). As the identification of cheaters does not reveal any information to the adversary, Multiplication allows the secure computation of the product of two shared secret values.

**Protocol Multiplication**$(\mathcal{P}, \mathcal{Z}, \mathbb{S}, [a], [b])$

```
0: Set $M = \emptyset$.
1: Invoke OptimisticMult$(\mathcal{P}, \mathcal{Z}, \mathbb{S}, [a], [b], Z)$ to compute
   $([c_1^{(Z)}], \ldots, [c_n^{(Z)}])$ and $[c^{(Z)}]$ for each $Z \in \mathcal{Z}$.
2: Set $\mathcal{Z}_M := \{Z \in \mathcal{Z} \mid M \subseteq Z\}$, fix some $\widetilde{Z} \in \mathcal{Z}_M$ and reconstruct the
   differences $[c^{(\widetilde{Z})}] - [c^{(Z)}] \ \forall Z \in \mathcal{Z}_M$.
3: If all differences are zero, output $[c^{(\widetilde{Z})}]$ as sharing of the product.
   Otherwise let $([d_1], \ldots, [d_n]) := ([c_1^{(\widetilde{Z})}], \ldots, [c_n^{(\widetilde{Z})}])$, $([e_1], \ldots, [e_n]) :=$
   $([c_1^{(Z)}], \ldots, [c_n^{(Z)}])$, $D := I_{\widetilde{Z}}$ and $E := I_Z$, where $[c^{(\widetilde{Z})}] - [c^{(Z)}] \neq 0$.
   a) Each $P_i$ shares the $2n$ values $d_{i,j} = \sum_{(p,q) \in D(i) \cap E(j)} a_p b_q$ and
      $e_{i,j} = \sum_{(p,q) \in E(i) \cap D(j)} a_p b_q$
   b) For each party $P_i$ reconstruct the differences $[d_i] - \sum_{j=1}^n [d_{i,j}]$
      and $[e_i] - \sum_{j=1}^n [e_{i,j}]$. If one of them is non-zero set
      $M \leftarrow M \cup \{P_i\}$ and continue at Step 2.
   c) For each (ordered) pair $(P_i, P_j)$ of parties reconstruct the
      difference $[d_{i,j}] - [e_{j,i}]$. If it is non-zero, reconstruct $[d_{i,j}]$,$[e_{j,i}]$
      and all shares $\{a_p, b_q \mid (p,q) \in D(i) \cap E(j)\}$ to find the cheater
      $P \in \{P_i, P_j\}$. Set $M \leftarrow M \cup \{P\}$ and continue at Step 2.
```

**Lemma 35.** *If $\mathbb{S}$ and $\mathcal{Z}$ satisfy $\mathcal{Q}^2(\mathbb{S}, \mathcal{Z})$ the protocol Multiplication yields a sharing $[c] = [ab]$. No information is leaked to the adversary. Multiplication communicates at most $\mathcal{O}(|\mathbb{S}||\mathcal{Z}|n^3 \log |\mathbb{F}| + |\mathbb{S}|n^5 \log |\mathbb{F}|)$ bits and broadcasts at most $\mathcal{O}(|\mathbb{S}||\mathcal{Z}|(n \log |\mathbb{F}| + n^2) + |\mathbb{S}|(n^3 \log |\mathbb{F}| + n^4))$ bits.*

*Proof. Correctness*: By invoking OptimisticMult for each $Z \in \mathcal{Z}$ it holds for $Z^*$ that $[c^{(Z^*)}] = [ab]$ (due to $\mathcal{Q}^2(\mathbb{S}, \mathcal{Z})$ $\forall S_p, S_q \in \mathbb{S} \ : \ S_p \cap S_q \setminus Z \neq \emptyset$ holds). If for every $Z \in \mathcal{Z}_M$ the difference in Step 2 is zero, then $[c^{(Z)}] = [ab]$ $\forall Z \in \mathcal{Z}_M$ ($M = \emptyset$ at the beginning). Hence the protocol terminates successfully outputting a sharing of $ab$. Otherwise there exists $[c^{(\widetilde{Z})}] - [c^{(Z)}] \neq 0$ and thus $\sum_{i=1}^n [d_i] \neq \sum_{i=1}^n [e_i]$. In Step 3a) each party is supposed to share a partition of his shares. Hence one of the following cases must occur: There exists a party $P_i$ such that $[d_i] \neq \sum_{j=1}^n [d_{i,j}]$ or $[e_i] \neq \sum_{j=1}^n [e_{i,j}]$. Or there exists a pair of parties $(P_i, P_j)$ such that $[d_{i,j}] \neq [e_{j,i}]$. In the first case $P_i$ will be detected as cheater in Step 3b). In the second case the cheater will be detected in Step 3c). In both cases $M \subseteq \mathcal{P}$ is strictly increased, hence the protocol will terminate after at most $n$ iterations. It holds that $M \subseteq Z^*$ and thus $Z^* \in \mathcal{Z}_M$. Therefore the correct sharing $[c^{(Z^*)}]$ is always used in Step 2 and the protocol will output the correct result.

*Privacy*: By the properties of the sharing scheme and Lemma 34 the invocation of Share, Reconstruct, OptimisticMult does not violate privacy. The adversary learns the differences reconstructed in Steps 2 and 3 of Multiplication, which are all zero unless the adversary cheats. In case of cheating the reconstructed values depends solely on the inputs of the adversary and are thus already known to him, thus privacy is not violated. All values further reconstructed in Step 3c) are known to the adversary before, as either $P_i$ or $P_j$ is corrupted.

*Complexity*: Follows from Lemmas 30, 33 and 34 by counting the number of invocations of the corresponding sub-protocols. $\square$

## 5.3.3 MPC Protocol

Combining Share, Reconstruct, and Multiplication the parties can securely compute a circuit $\mathcal{C}$ over $\mathbb{F}$, where all intermediate values are shared according to Definition 27.

**Protocol MPC($\mathcal{P}, \mathcal{Z}, \mathcal{C}$)**

0: The parties take $\mathbb{S} := \{\mathcal{P} \setminus Z | Z \in \mathcal{Z}\}$ as sharing specification.

1: For every gate of $C$ being evaluated do the following:

- *Input gate for $P_D$:* Share($\mathcal{P}, \mathcal{Z}, \mathbb{S}, P_D, s$) is invoked to share $s$, where $P_D$ is the input-giving party.
- *Linear gate:* The linear combination of the corresponding shares is computed locally using the linearity of the sharing.
- *Random gate:* Each party shares a random value. The sum of these values is used as output of the gate.
- *Multiplication gate:* Multiplication($\mathcal{P}, \mathcal{Z}, \mathbb{S}, [a], [b]$) is used to multiply $[a]$ and $[b]$.
- *Output gate:* The parties invoke Reconstruct($\mathcal{P}, \mathcal{Z}, \mathbb{S}, [s], R$) to reconstruct the sharing $[s]$ to parties in $R$.

**Theorem 7.** *Let $\mathcal{P}$ be a set of $n$ parties, $\mathcal{C}$ a circuit over $\mathbb{F}$ and $\mathcal{Z}$ an adversary structure satisfying $Q^3(\mathcal{P}, \mathcal{Z})$, then MPC($\mathcal{P}, \mathcal{Z}, \mathcal{C}$) perfectly*

*$\mathcal{Z}$-securely evaluates $\mathcal{C}$. It communicates $|\mathcal{C}||\mathcal{Z}|^2 \cdot \mathrm{poly}(n, \log|\mathbb{F}|)$ bits.*

*Proof.* It is easy to see that $\mathbb{S} := \{\mathcal{P} \setminus Z | Z \in \mathcal{Z}\}$ is a sharing specification satisfying $\mathcal{Q}^2(\mathbb{S}, \mathcal{Z})$. Hence by the properties of the sharing scheme and Lemma 35 the statement follows. The protocol communicates in total $\mathcal{O}(|\mathcal{C}||\mathcal{Z}|^2 n^3 \log|\mathbb{F}| + |\mathcal{C}||\mathcal{Z}|n^5 \log|\mathbb{F}|)$ bits and broadcasts $\mathcal{O}(|\mathcal{C}||\mathcal{Z}|^2(n \log|\mathbb{F}| + n^2) + |\mathcal{C}||\mathcal{Z}|(n^3 \log|\mathbb{F}| + n^4))$ bits. Broadcast can be simulated with the protocol in [FM98], which communicates poly($n$) bits in order to broadcast one bit. This yields the claimed communication complexity. □

## 5.4 Unconditional Protocol

Our main result is an MPC protocol unconditionally $\mathcal{Z}$-secure for an $\mathcal{Q}^2$ adversary structure $\mathcal{Z}$. Its communication complexity is linear in $|\mathcal{Z}|$. This is the first protocol reaching the optimal lower bound of $\Omega(|\mathcal{Z}|)$ on the computational complexity (see Section 5.6).

### 5.4.1 Information Checking

In the perfect model, $\mathcal{Q}^3$ enables the honest parties to securely reconstruct shares, as it assures that every share is held by enough honest parties. Here, $\mathcal{Q}^2$ only ensures that each share is held by at least one honest party. Correctness is achieved by the use of information checking, a technique that prevents (malicious) parties from announcing wrong values (see [RB89, Bea91, CDD+99, HMZ08]). The following information-checking protocol is a slight variation of [CDD+99]. It is a three party protocol between a sender $P_i$, a recipient $P_j$ and a verifier $P_k$. The sender $P_i$ provides $P_j$ with some authentication tag and $P_k$ with some verification tag, such that $P_j$ later can prove the authenticity of a value $s$ to the verifier $P_k$. We assume that each pair $P_i, P_k$ of parties knows a fixed secret value $\alpha_{i,k} \in \mathbb{F} \setminus \{0, 1\}$.

**Definition 28.** *A vector $(s, y, z, \alpha)$ is 1-consistent if there exists a polynomial $f$ of degree 1 over $\mathbb{F}$ such that $f(0) = s, f(1) = y, f(\alpha) = z$. We say a value $s$ is $(P_i, P_j, P_k)$-*authenticated *if $P_j$ knows $s$ and some authentication tag $y$ and $P_k$ knows a verification tag $z$ such that $(s, y, z, \alpha_{i,k})$ is 1-consistent. The vector $(y, z, \alpha_{i,k})$ is denoted by $A_{i,j,k}(s)$.*

**Lemma 36.** *A $(P_i, P_j, P_k)$-authenticated value $s$ does not leak information to $P_k$.*

*Proof.* The verification tag $z$ is statistically independent of value $s$. □

**Lemma 37.** *Let $s$ be $(P_i, P_j, P_k)$-authenticated, i.e. $(s, y, z, \alpha_{i,k})$ is 1-consistent. Then for $P_j$ being able to find an authentication tag $y'$ for a value $s' \neq s$ such that $(s', y', z, \alpha_{i,k})$ is 1-consistent is equivalent to finding $\alpha_{i,k}$.*

*Proof.* If both $(s, y, z, \alpha_{i,k})$ and $(s', y', z, \alpha_{i,k})$ are 1-consistent, then also $(s - s', y - y', 0, \alpha_{i,k})$ is 1-consistent. The corresponding polynomial of degree 1 is not parallel to the $x$-axis, as $s - s' \neq 0$. Thus it has an unique root at $\alpha_{i,k} = \frac{s-s'}{s-s'-y+y'}$. □

**Lemma 38.** *The parties $P_j$ and $P_k$ can locally compute an authentication and a verification tag of any linear combination of $(P_i, P_j, P_k)$-authenticated values (for fixed $P_i$). This is called the linearity of the authentication.*

*Proof.* Let $s_a$ and $s_b$ be $(P_i, P_j, P_k)$-authenticated with authentication tags $y_a, y_b$ and verification tags $z_a, z_b$ and the (fixed) point $\alpha_{i,k}$ and let $L$ be a linear function. Then $L(s_a, s_b)$ is $(P_i, P_j, P_k)$-authenticated with authentication tag $y = L(y_a, y_b)$ and verification tag $z = L(z_a, z_b)$. This works as the polynomials of degree 1 over $\mathbb{F}$ form a vector space, hence $(L(s_a, s_b), L(y_a, y_b)\ , L(z_a, z_b), \alpha_{i,k})$ is 1-consistent. □

Let $s$ be a value known to $P_j$ and $P_k$. Then these parties can use the protocol DefaultAuthenticate to $(P_i, P_j, P_k)$-authenticate $s$ without communication for arbitrary $P_i$. Note that $P_i$ does not play an (active) role in this protocol.

**Protocol DefaultAuthenticate**$(P_i, P_j, P_k, s)$

```
0: $P_j, P_k$ take the value $s$ as input.
1: $P_j$ outputs authentication tag $y = s$. $P_k$ outputs verification tag
   $z = s$.
```

**Lemma 39.** *If the value $s$ is known to the honest parties in $\{P_j, P_k\}$ protocol DefaultAuthenticate$(P_i, P_j, P_k, s)$ securely $(P_i, P_j, P_k)$-authenticates $s$ without any communication.*

*Proof. Correctness:* $(s, s, s, \alpha_{i,k})$ is 1-consistent for any $\alpha_{i,k}$.

*Privacy and Communication:* No communication occurs and the adversary does not learn new information. □

The non-robust protocol Authenticate allows to securely $(P_i, P_j, P_k)$-authenticate a (secret) value $s$.

**Protocol Authenticate**$(P_i, P_j, P_k, s)$

```
0: $P_i$ and $P_j$ take the value $s$ as input.
1: $P_i$ chooses random values $(y, z) \in \mathbb{F}$ such that $(s, y, z, \alpha_{i,k})$ is
   1-consistent and random values $(s', y', z') \in \mathbb{F}$ such that
   $(s', y', z', \alpha_{i,k})$ is 1-consistent and sends $(s', y, y')$ to party $P_j$ and
   $(z, z')$ to party $P_k$.
2: $P_k$ broadcasts random $r \in \mathbb{F}$.
3: $P_i$ broadcasts $s'' = rs + s'$ and $y'' = ry + y'$.
4: $P_j$ checks if $s'' = rs + s'$ and $y'' = ry + y'$ and broadcast OK or
   NOK accordingly. If NOK was broadcast the protocol is aborted.
5: $P_k$ checks if $(s'', y'', rz + z', \alpha_{i,k})$ is 1-consistent. If yes $P_k$ sends OK
   to $P_j$ otherwise he sends $(\alpha_{i,k}, z)$ to $P_j$, who adjusts $y$ such that
   $(s, y, z, \alpha_{i,k})$ is 1-consistent.
6: $P_j$ outputs $y$ and $P_k$ outputs $z$.
```

**Lemma 40.** *If $P_k$ is honest and $s$ is known to the honest parties in $\{P_i, P_j\}$. Then the protocol* Authenticate$(P_i, P_j, P_k, s)$ *either securely $(P_i, P_j, P_k)$-authenticates $s$ or aborts except with error probability of at most $\frac{1}{|\mathbb{F}|}$. In the case of an abort a party in $\{P_i, P_j\}$ is corrupted. The protocol communicates at most* $7 \log |\mathbb{F}|$ *bits and broadcasts at most* $3 \log |\mathbb{F}| + 1$ *bits.*

*Proof. Correctness:* If the protocol was aborted, either $s'' \neq rs + s'$ or $y'' \neq ry + y'$ meaning $P_i$ is corrupted, or $P_j$ misleadingly accused $P_i$. Otherwise, the parties use some $(s, y, z, \alpha_{i,k})$ as authentication of $s$. The probability that $(s, y, z, \alpha_{i,k})$ is not 1-consistent is $|\mathbb{F}|^{-1}$, as for a fixed $r$ there is exactly one way to choose $y, z$ such that the inconsistency is not detected.

*Privacy:* The verification tag $z$, the values $s''$ and $y''$ are statistically independent of the value $s$. Also $\alpha_{i,k}$ is sent only to $P_j$ if either $P_i$ or $P_k$ is malicious.

*Communication:* Seen by counting the number of messages sent or broadcast during the protocol. □

**Remark.** *If the (honest) parties $P_i$ and $P_j$ do not know the same $s$ the protocol will abort as well.*

Assume that $P_k$ knows a candidate $s'$ for a $(P_i, P_j, P_k)$-authenticated value $s$. If $P_j$ wants to prove the authenticity of $s'$ (i.e. that $s' = s$) the parties invoke the protocol Verify. If $P_k$ accepts the proof he outputs $s'$, otherwise he outputs $\perp$.

**Protocol Verify**$(P_i, P_j, P_k, s', A_{i,j,k}(s))$

```
0: Let $A_{i,j,k}(s) = (y, z, \alpha_{i,k})$. $P_j$ takes $y$ as input and $P_k$ takes $s', z$ as
   input.
1: $P_j$ sends $y$ to $P_k$
2: $P_k$ outputs $s'$ if $(s', y, z, \alpha_{i,k})$ is 1-consistent otherwise $\perp$.
```

**Lemma 41.** *Assume $s$ is $(P_i, P_j, P_k)$-authenticated and let $P_k$ be an honest party knowing $s'$. If $P_j$ is honest and $s' = s$, $P_k$ will output $s$ in Verify. Otherwise $P_k$ will output $\perp$ or $s$ except with error probability of at most $\frac{1}{|\mathbb{F}|-2}$. The protocol communicates at most* $\log|\mathbb{F}|$ *bits.*

*Proof. Correctness:* Let $P_k$ be an honest party,let $A_{i,j,k}(s) = (y, z, \alpha_{i,k})$ be consistent with $s$, i.e. $(s, y, z, \alpha_{i,k})$ is 1-consistent and assume that $s' = s$. If $P_j$ sends the right $y$ the vector $(s', y, z, \alpha_{i,k})$ is 1-consistent and $P_k$ will output $s$. Otherwise $P_k$ always outputs $\perp$. So assume $s' \neq s$. Then the probability of finding $y'$ such that the vector $(s', y', z, \alpha_{i,k})$ is 1-consistent is at most $\frac{1}{|\mathbb{F}|-2}$, thus $P_k$ outputs $\perp$ except with error probability of at most $\frac{1}{|\mathbb{F}|-2}$.

*Privacy and Communication:* No information except $y$ is sent. □

## 5.4.2 Unconditional Secret Sharing

Starting from the secret sharing of Section 5.3.1 we construct a sharing scheme for the $\mathcal{Q}^2$ case using the information-checking scheme of the previous section.

**Definition 29.** *A value $s$ is* shared *with respect to the sharing specification $\mathbb{S} = (S_1, \ldots, S_h)$, if the following holds:*

a) *There exist shares $s_1, \ldots, s_h$ such that $s = \sum_{q=1}^{h} s_q$*

b) *Each $s_q$ is known to every (honest) party in $S_q$*

c) *$\forall P_i, P_j \in S_q$ $P_k \in \mathcal{P}$ $s_q$ is $(P_i, P_j, P_k)$-authenticated.*

We denote the sharing of a value $s$ by $[s]$. Let $[s]_q = (s_q, \{A_{i,j,k}(s_q)\})$, where $s_q$ is the $q$-th share and $\{A_{i,j,k}(s_q)\}$ the set of all associated authentications. As the perfect sharing from Section 5.3.1 this sharing is linear and does not leak information to the adversary (for a $\mathcal{Z}$-private $\mathbb{S}$).

The following protocol allows a dealer $P_D$ to securely share a secret value $s$.

**Protocol Share**$(\mathcal{P}, \mathcal{Z}, \mathbb{S}, P_D, s)$

```
0: The dealer $P_D$ takes $s$ as input.
1: $P_D$ splits $s$ into random shares $s_1, \ldots, s_{|\mathbb{S}|}$ subject to $s = \sum_{q=1}^{|\mathbb{S}|} s_q$.
2: **for all** $q \in \{1, \ldots, |\mathbb{S}|\}$ **do**
3:     $P_D$ sends share $s_q$ to every party in $S_q$.
4:     $\forall P_i, P_j \in S_q$ and $\forall P_k \in \mathcal{P}$ invoke Authenticate$(P_i, P_j, P_k, s_q)$.
       If (for fixed $q$) any Authenticate$(P_i, P_j, P_k, s_q)$ was aborted
       $P_D$ broadcasts $s_q$, the parties in $S_q$ replace there share and
       DefaultAuthenticate$(P_i, P_j, P_k, s_q)$ is invoked
   $\forall P_i, P_j \in S_q$ $\forall P_k \in \mathcal{P}$.
5: **end for**
6: The parties in $\mathcal{P}$ collectively output $[s]$.
```

**Lemma 42.** *For any adversary structure $\mathcal{Z}$ protocol Share$(\mathcal{P}, \mathcal{Z}, \mathbb{S}, P_D, s)$ securely computes a sharing $[s']$ except with error probability of at most $\frac{1}{|\mathbb{F}|} n^3 |\mathbb{S}|$ and if $P_D$ is honest $s' = s$. The protocol communicates at most $|\mathbb{S}|(7n^3 + n) \log |\mathbb{F}|$ bits and broadcasts at most $|\mathbb{S}|((3n^3 + 1) \log |\mathbb{F}| + n^3)$ bits.*

*Proof. Correctness:* Assume that $P_D$ does not send the same value to the (honest) parties in $S_q$ (Step 3). In this case at least one invocation of Authenticate will abort (see Remark 5.4.1) and $P_D$ must broadcast the value. Otherwise all (honest) parties use the same value $s_q$ in Step 3. We have to show that every (honest) $P_j$ gets his authentications $A_{i,j,k}(s_q)$. If all instances of Authenticate do not abort the statement follows from Lemma 40. Otherwise $s_q$ is broadcast and the parties use

DefaultAuthenticate resulting in the proper sharing state (c.f. Lemma 39). Note that a single invocation of Authenticate has an error probability of at most $\frac{1}{|\mathbb{F}|}$, so the above upper bound on the error probability follows.

*Privacy:* We only have to check that broadcasting $s_q$ in Step 4 does not violate privacy. But $s_q$ is only broadcast when at least one Authenticate was aborted. In this case either $P_D$ or a party in $S_q$ is malicious, hence $s_q$ is known to the adversary before the broadcast (Lemma 40 and Remark 5.4.1).

*Communication:* Follows directly by counting the numbers of messages sent or broadcast (c.f. Lemmas 40 and 39) □

If a value is publicly known the party can use DefaultShare to obtain a sharing of it.

**Protocol DefaultShare**$(\mathcal{P}, \mathcal{Z}, \mathbb{S}, s)$

```
0: Every party takes $s$ as input.
1: The share $s_1$ is set to $s$ and all other shares are set to 0.
2: DefaultAuthenticate$(P_i, P_j, P_k, s_q)$ is invoked
   $\forall S_q \forall P_i, P_j \in S_q \; \forall P_k \in \mathcal{P}$.
3: The parties in $\mathcal{P}$ collectively output $[s]$.
```

**Lemma 43.** *DefaultShare$(\mathcal{P}, \mathcal{Z}, \mathbb{S}, s)$ securely computes a sharing $[s]$ of $s$. The protocol does not communicate.*

*Proof.* The statement follows from Lemmas 31 and 39. □

The protocol ReconstructShare allows reconstruction of a share from some sharing $[s]$ to parties in $R \subseteq \mathcal{P}$. Hence the parties can reconstruct $s$ by invoking protocol ReconstructShare for each share of $[s]$.

**Protocol ReconstructShare**$(\mathcal{P}, \mathcal{Z}, \mathbb{S}, [s]_q, R)$

```
0: The parties in $S_q$ take collectively $[s]_q = (s_q, \{A_{i,j,k}(s_q)\})$ as input.
1: Every party $P_j$ in $S_q$ sends $s_q$ to every party in $R$.
2: for all $P_j \in S_q, P_k \in R$ do
3:     Invoke Verify$(P_i, P_j, P_k, s_q^{(j)}, A_{i,j,k}(s_q))$ $\forall P_i \in S_q$ where $s_q^{(j)}$ is
       the value received by $P_k$ from $P_j$ in Step 1. If $P_k$ output $s_q^{(j)}$ in
```

```
         each invocation he accepts it as value for $s_q$.
4: **end for**
5: Each $P_k$ outputs some value he accepted in Step 3 (or $\perp$ if never
   accepted a value).
```

**Lemma 44.** *Assume $S_q$ and $\mathcal{Z}$ satisfy $\mathcal{Q}^1(S_q, \mathcal{Z})$ and let $[s]_q$ be a consistent share. Every honest party in $R$ outputs $s_q$ in* ReconstructShare *except with error probability of at most $\frac{1}{|\mathbb{F}|-2} n|S_q|$. The protocol communicates at most $(n^3 + n^2) \log |\mathbb{F}|$ bits and does not broadcast.*

*Proof. Correctness:* As $S_q$ and $\mathcal{Z}$ satisfy $\mathcal{Q}^1(S_q, \mathcal{Z})$ there exists at least one honest party $P_j$ in $S_q$, who sends the right value $s_q$ to $P_k \in R$ in Step 1. Hence every (honest) $P_k$ will accept $s_q$ in Step 3 from $P_j$, as $P_j$ has a valid authentication for $s_q$ from every party in $S_q$ (c.f. Lemma 41). On the other hand a malicious party does not have a valid authentication for $s'_q \neq s_q$ from every party in $S_q$ (one of them is honest!). So no honest party will accept $s'_q \neq s_q$ in Step 3 and thus $P_k$ output $s_q$ in the last step except with error probability of at most $\frac{1}{|\mathbb{F}|-2}|S_q|$ (c.f. Lemma 41). As there are at most $n$ parties in $R$ the overall error probability follows.

*Privacy:* Follows from Lemma 41.

*Communication:* Follows directly by counting the numbers of messages sent (c.f. Lemma 41) □

**Protocol Reconstruct**$(\mathcal{P}, \mathcal{Z}, \mathbb{S}, [s], R)$

```
0: The parties in $\mathcal{P}$ take collectively $[s]$ as input.
1: **for all** $q = 1, \ldots, |\mathbb{S}|$ **do**
2:     ReconstructShare$(\mathcal{P}, \mathcal{Z}, \mathbb{S}, [s]_q, R)$ is invoked.
3: **end for**
4: The parties locally sum up the shares to obtain and output $s$.
```

**Lemma 45.** *Assume $\mathbb{S}$ and $\mathcal{Z}$ satisfy $\mathcal{Q}^1(\mathbb{S}, \mathcal{Z})$ and let $[s]$ be a sharing of the value $s$. Every honest party in $R$ outputs $s$ in* Reconstruct *except with error probability of at most $\frac{1}{|\mathbb{F}|-2} n^2|\mathbb{S}|$. The protocol communicates at most $|\mathbb{S}|(n^3 + n^2) \log |\mathbb{F}|$ bits and does not broadcast.*

*Proof.* The statement follows directly from Lemma 44, as the parties invoke the protocol ReconstructShare for each share. □

### 5.4.3 Multiplication

We present a protocol for the unconditionally-secure computation of the (shared) product of two shared values $[a]$ and $[b]$. The idea is, as in the perfect case, to use an optimistic multiplication. The protocol BasicMult takes a set $M$ of (identified) malicious parties as input and outputs the correct product given that no party in $\mathcal{P} \setminus M$ actively cheated. In a next step a probabilistic check is used to determine whether the product computed in BasicMult is correct. This allows us to detect malicious behavior. If cheating occurred, all involved sharings (from BasicMult) are reconstructed to identify a cheater in $\mathcal{P} \setminus M$. These reconstructions violate the privacy of the involved factors if the protocol is not used directly in the actual circuit computation. Instead we use it to multiply two random values and make use of circuit randomization from [Bea92] for actual multiplication gates.

**Protocol BasicMult**$(\mathcal{P}, \mathcal{Z}, \mathbb{S}, [a], [b], M)$

```
0: The parties in $\mathcal{P}$ take collectively $[a], [b]$ and $M$ as input.
1: $\forall S_q : S_q \cap M \neq \emptyset$ invoke ReconstructShare to reconstruct $a_q$ and $b_q$.
2:   a) Each party $P_i \in \mathcal{P} \setminus M$ (locally) computes his designated
        products and shares the sum $c_i = \sum_{(p,q) \in I(i)} a_p b_q$.
     b) For each $P_i \in M$ DefaultShare$(\mathcal{P}, \mathcal{Z}, \mathbb{S}, c_i)$ is invoked where
        $c_i = \sum_{(p,q) \in I(i)} a_p b_q$.
3: The parties collectively output $([c_1], \ldots, [c_n])$ and $[c] = \sum_{i=1}^{n} [c_i]$.
```

**Lemma 46.** *Let $M \subseteq Z^*$ be a set of (identified) malicious parties and assume that $\mathcal{Z}$ and $\mathbb{S}$ satisfy $\mathcal{Q}^1(\mathbb{S}, \mathcal{Z})$. Then BasicMult$(\mathcal{P}, \mathcal{Z}, \mathbb{S}, [a], [b], M)$ securely computes sharings $[c], ([c_1], \ldots, [c_n])$ except with error probability of $\mathcal{O}(\frac{1}{|\mathbb{F}|} n^4 |\mathbb{S}|)$. If no party in $\mathcal{P} \setminus M$ actively cheats, it holds that $\forall i\ c_i = \sum_{(p,q) \in I_Z(i)} a_p b_q$ and $c = ab$. The protocol communicates at most $\mathcal{O}(|\mathbb{S}| n^4 \log |\mathbb{F}|)$ bits and broadcasts at most $\mathcal{O}(|\mathbb{S}| n^4 \log |\mathbb{F}|)$ bits.*

*Proof. Correctness:* The properties of the sharing protocol guarantee that the outputs are valid sharings except with error probability of $\mathcal{O}(\frac{1}{|\mathbb{F}|} n^4 |\mathbb{S}|)$. The $\mathcal{Q}^1(\mathbb{S}, \mathcal{Z})$ property allows the parties to securely reconstruct shares and grants that there exists a proper assignment of parties in $\mathcal{P}$ to the local products. If none of the parties in $\mathcal{P} \setminus M$ cheated, it holds for each

$P_i$ that $c_i = \sum_{(p,q)\in I_Z(i)} a_p b_q$ (for parties in $M$ DefaultShare is used on reconstructed values).

*Privacy:* All reconstructed shares $a_q$,$b_q$ are known to parties in $M$.

*Complexity:* Follow directly from the properties of the sharing scheme (c.f. Lemmas 42, 43 and 44). □

## Detectable Random Triple Generation

The following unconditionally secure protocol takes a set $M$ of malicious parties as an additional input and computes a random multiplication triple $([a],[b],[c])$ where $c = ab$ given that no party in $\mathcal{P} \setminus M$ actively cheats. Otherwise it outputs a set of malicious parties $M'$ such that $M \subsetneq M'$. This protocol uses a probabilistic check to detect cheating. First the parties generate a shared random challenge $[r]$ and a blinding $[b']$. Then they use BasicMult to compute the sharings $[c] = [a][b]$, $[c'] = [a][b']$ and check whether $[a](r[b] + [b']) = (r[c] + [c'])$. If this is the case the multiplication triple $([a],[b],[c])$ is output. Otherwise the parties identify (at least) one cheater in $\mathcal{P} \setminus M$ by reconstructing $[a],[b],[b'],[c],[c']$.

**Lemma 47.** *If* $\mathbb{S}$ *and* $\mathcal{Z}$ *satisfy* $\mathcal{Q}^1(\mathbb{S},\mathcal{Z})$ *and* $M \subseteq Z^*$*, the protocol RandomTriple outputs either a random multiplication triple* $([a],[b],[c])$ *or set* $M' \subseteq Z^*$ *where* $M \subsetneq M'$ *except with error probability of* $\mathcal{O}(\frac{1}{|\mathbb{F}|}|\mathbb{S}|n^4) + \frac{1}{|\mathbb{F}|}$*. No information is leaked to the adversary. RandomTriple communicates at most* $\mathcal{O}(|\mathbb{S}|n^4 \log|\mathbb{F}|)$ *bits and broadcasts at most* $\mathcal{O}(|\mathbb{S}|n^4 \log|\mathbb{F}|)$ *bits.*

*Proof. Correctness:* In Step 2, the parties compute $[c]$ and $[c']$. Given that no party in $\mathcal{P} \setminus M$ actively cheated it holds that $c = ab$ and $c' = ab'$. In this case $[a](r[b] + [b']) - r[c] - [c']$, which is computed in Step 3, is zero for all $r$ and the parties reconstruct the random multiplication triple $([a],[b],[c])$. If $c \neq ab$ the difference $[a](r[b] + [b']) - r[c] - [c']$ is non-zero except for at most one $r$ and the parties go to Step 5 with probability at least $(1 - \frac{1}{|\mathbb{F}|})$ (assuming that no errors happen in sharing and reconstruction of values). For at least one party $P_i \in \mathcal{P} \setminus M$ it must hold that $rc_i + c'_i \neq \sum_{(p,q)\in I(i)} r(a_p b_q) + (a_p b'_q)$. By opening all involved sharing it is easy to find these parties. Thus it holds that $M \subsetneq M'$ and $M' \subseteq Z^*$. The overall error probability is composed of the error probability of the sharing scheme and the one of the random challenge check in Step 3.

*Privacy:* Neither the protocol BasicMult nor the sharing scheme do violate privacy (c.f. Lemma 46). The values $e$ is statistically independent of $([a], [b], [c])$, as $b'$ acts as blinding. If no cheating occurred the value $d$ is always zero. If Step 5 is invoked, the reconstructed values are not used, and privacy is met.

*Communication:* Follows from counting the number of messages sent (c.f. Lemmas 42, 45 and 46). □

**Protocol RandomTriple**$(\mathcal{P}, \mathcal{Z}, \mathbb{S}, M)$

```
0: The parties take the set $M \subseteq \mathcal{P}$ as input.
1: The parties generate random shared values $[a], [b], [b'], [r]$ by
   summing up shared random values (one from each party) for each
   value.
2: Invoke BasicMult$(\mathcal{P}, \mathcal{Z}, \mathbb{S}, [a], [b], M)$ to compute the sharing $[c]$ and
   the vector $([c_1], \dots, [c_n])$ and invoke BasicMult$(\mathcal{P}, \mathcal{Z}, \mathbb{S}, [a], [b'], M)$
   to compute the sharing $[c']$ and the vector $([c'_1], \dots, [c'_n])$.
3: Reconstruct $[r]$ and (locally) compute $[e] = r[b] + [b']$ and
   reconstruct it to obtain $e$. Then $[d] = e[a] - r[c] - [c']$ is computed
   (locally) and reconstructed.
4: If the value $d$ is zero the parties output $([a], [b], [c])$.
5: Otherwise reconstruct the sharings $[a], [b], [b'], [c_1], \dots, [c_n], [c'_1], \dots,$
   $[c'_n]$. The parties output
   $M' = M \cup \{P_i : rc_i + c'_i \neq \sum_{(p,q) \in I(i)} r(a_p b_q) + (a_p b'_q)\}$.
```

## Multiplication with Circuit Randomization

The actual multiplication is based on circuit randomization [Bea92]. It allows parties to compute the product $[xy]$ of two shared values $[x]$ and $[y]$ at the cost of two reconstructions given a random multiplication triple $([a], [b], [c])$, where $ab = c$. The trick is to use that $xy = ((x-a)+a)((y-b)+b)$. By reconstructing $d = x - a$ and $e = y - b$ the parties can compute $[xy]$ as $de + d[b] + [a]e + [c]$. This does not violate the secrecy of $[x]$ or $[y]$ as the random values $[a]$ and $[b]$ act as blinding.

**Protocol Multiplication**$(\mathcal{P}, \mathcal{Z}, \mathbb{S}, [x], [y])$

```
0: The parties in $\mathcal{P}$ take collectively $[x], [y]$ as input and set $M := \emptyset$.
1: Invoke RandomTriple$(\mathcal{P}, \mathcal{Z}, \mathbb{S}, M)$. If the protocol outputs a set $M'$,
   set $M \leftarrow M'$ and repeat Step 1. Otherwise use the output as
   random multiplication triple $([a], [b], [c])$.
2: Compute and reconstruct $[d_x] = [x] - [a]$ and $[d_y] = [y] - [b]$.
   Compute $d_x d_y + d_x[b] + d_y[a] + [c] = [xy]$ to obtain a sharing of $xy$.
```

**Lemma 48.** *Multiplication$(\mathcal{P}, \mathcal{Z}, \mathbb{S}, [x], [y])$ is an unconditional secure multiplication protocol given that $\mathbb{S}$ and $\mathcal{Z}$ satisfy $\mathcal{Q}^1(\mathbb{S}, \mathcal{Z})$. The protocol has an error probability of $\mathcal{O}(\frac{1}{|\mathbb{F}|}|\mathbb{S}|n^5)$. The protocol communicates at most $\mathcal{O}(|\mathbb{S}|n^5 \log |\mathbb{F}|)$ bits and broadcasts at most $\mathcal{O}(|\mathbb{S}|n^5 \log |\mathbb{F}|)$ bits.*

*Proof. Correctness:* Assume that RandomTriple in Step 1 outputs a set $M'$, then we have that $M \subsetneq M' \subseteq \mathcal{P}$. Hence this step is repeated less then $n$ times and results in a random multiplication triple $([a], [b], [c])$ (c.f. Lemma 47). The rest of the protocol is just the multiplication from [Bea92].

*Privacy:* Follows from [Bea92] and Lemma 47.

*Communication:* Follows from counting the number of messages sent (cf. Lemmas 45 and 47). □

### 5.4.4 Unconditional MPC Protocol

The combination of the protocols Share, Reconstruct, and Multiplication leads directly to the following unconditionally secure MPC protocol which secure against adversaries satisfying $\mathcal{Q}^2$.

**Protocol MPC**$(\mathcal{P}, \mathcal{Z}, \mathcal{C})$

```
0: The parties take $\mathbb{S} := \{\mathcal{P} \setminus Z | Z \in \mathcal{Z}\}$ as sharing specification.
1: For every gate of $\mathcal{C}$ being evaluated do the following:
      - Input gate for $P_D$: Share$(\mathcal{P}, \mathcal{Z}, \mathbb{S}, P_D, s)$ is invoked to share $s$
      - Linear gate: The linear combination of the corresponding
        shares is computed locally using the linearity of the sharing.
      - Random gate: Each party shares a random value. The sum of
        these values is used as output of the gate.
```

- *Multiplication gate:* Multiplication($\mathcal{P}, \mathcal{Z}, \mathbb{S}, [x], [y]$) is used to multiply $[x]$ and $[y]$.
- *Output gate:* The parties invoke Reconstruct($\mathcal{P}, \mathcal{Z}, \mathbb{S}, [s], R$) to reconstruct $s$ for parties in $R$.

**Theorem 8.** *Let $\mathcal{C}$ be a circuit over $\mathbb{F}$, where $|F| \in \Omega(2^{\kappa})$ and $\kappa$ is a security parameter, and let $\mathcal{Z}$ be an adversary structure satisfying $\mathcal{Q}^2(\mathcal{P}, \mathcal{Z})$, then MPC($\mathcal{P}, \mathcal{Z}, \mathcal{C}$) $\mathcal{Z}$-securely evaluates $\mathcal{C}$ with an error probability of $2^{-\kappa}|\mathcal{C}||\mathcal{Z}| \cdot \mathrm{poly}(n, \kappa)$. It communicates $|\mathcal{C}||\mathcal{Z}| \cdot \mathrm{poly}(n, \kappa)$ bits and broadcasts $|\mathcal{C}||\mathcal{Z}| \cdot \mathrm{poly}(n, \kappa)$ bits within $\mathrm{poly}(n, \kappa) \cdot d$ rounds, where $d$ denotes the multiplicative depth of $C$.*

*Proof. Correctness:* It is easy to see that $\mathbb{S} := \{\mathcal{P} \setminus Z | Z \in \mathcal{Z}\}$ is a sharing specification satisfying $\mathcal{Q}^1(\mathbb{S}, \mathcal{Z})$. Hence by the properties of the sharing scheme and Lemma 48 correctness and the bound on the error probability follow.

*Communication:* The claimed communication and broadcast complexity follow directly from the used subprotocols. Inspection of the subprotocols also shows that it is possible to evaluate gates on the same multiplicative depth of $C$ in parallel. As each subprotocol only requires $\mathrm{poly}(n, \kappa)$ rounds, the total number of rounds follows. $\square$

Note that broadcast channels can be (unconditionally secure) emulated using the protocol from [PW96], which communicates $\mathrm{poly}(n, \kappa)$ bits in order to broadcast one bit (with error probability of $\mathcal{O}(2^{-\kappa})$). This results in an MPC protocol with the same efficiency and error probability as stated in Theorem 8.

The error probability of the presented protocol grows linearly in the size of the adversary structure $\mathcal{Z}$. As $|\mathcal{Z}|$ is typically exponential in $n$, the security parameter $\kappa$ must be chosen accordingly (such that $|\mathcal{Z}| \in \mathrm{poly}(\kappa)$). This results in a huge security parameter and therefore in inefficient protocols.

We therefore provide in the next section an extension of the previous protocol in which the error probability only depends on $\log |Z|$. In this extended version of the protocol a reasonably large security parameter $\kappa$ is sufficient.

## 5.5 Unconditional Protocol for Superpolynomial $|\mathcal{Z}|$

The protocol from the previous section has an error probability linear in $|\mathcal{Z}|$, which is problematic for large adversary structures $\mathcal{Z}$. In this section, we present modifications to the protocol that reduce the dependency to $\log |\mathcal{Z}|$, which is in $\text{poly}(n)$.

The reason for the error probability being dependent on $|\mathcal{Z}|$ is twofold: Firstly, the protocol requires $\Omega(|\mathcal{Z}|)$ probabilistic checks, in each of them a cheating party might remain undetected with probability $2^{-\kappa}$. Secondly, the protocol requires $\Omega(|\mathcal{Z}|)$ broadcasts, each of them having a small probability of failure.

### 5.5.1 Information Checking

In each invocation of Authenticate / Verify, a cheating attempt of a malicious party $P_i$ is not detected with probability of $\mathcal{O}(\frac{1}{|\mathbb{F}|})$ (c.f. Section 5.4.1). As these protocols are invoked $\Theta(|\mathcal{Z}|)$ times per sharing, the resulting error probability depends linearly on $|\mathcal{Z}|$. To avoid this we use local dispute control to deal with detected cheaters.

More formally, each party $P_k$ locally maintains a list $\mathcal{L}_k$ of parties whom he distrusts. At the beginning of the MPC protocol these lists are empty. Protocol Authenticate is modified, such that $P_j$ puts $P_i$ on his list $\mathcal{L}_j$ if the check in Step 4 fails. Once $P_i \in \mathcal{L}_j$, $P_j$ behaves in all future invocations of the protocol as if the check in Step 4 failed independently whether this is the case or not. Similarly $P_k$ puts $P_i$ on his list $\mathcal{L}_k$ if the check in Step 5 fails. As soon as $P_i \in \mathcal{L}_k$, $P_k$ behaves in Step 5 as if the corresponding check failed. Furthermore, in protocol Verify, $P_k$ puts $P_j$ on his list $\mathcal{L}_k$ if the check in Step 2 failed. Again $P_k$ behaves for all $P_j \in \mathcal{L}_k$ as if the check failed independently whether this is the case or not.

In both protocols the adversary has a chance of $\mathcal{O}(\frac{1}{|\mathbb{F}|})$ to cheat successfully, but if he fails (with probability $\Omega(1 - \frac{1}{|\mathbb{F}|})$) one corrupted party $P_i$ is put on the list $\mathcal{L}_k$ of an honest party $P_k$. From then on $P_i$ is never able to cheat in instances of both protocols when $P_k$ takes part (in the right position). This means that the adversary actually has at most $n^2$ attempts to cheat. Hence total error probability of arbitrary many instances of Verify and Authenticate is at most $\mathcal{O}(\frac{1}{|\mathbb{F}|}n^2)$ and no longer depends on $\mathcal{Z}$.

Note that the parallel invocation of Authenticate, as it is used in Share, requires special care. For example if in one of the parallel invocations of Authenticate (with $P_i$ and $P_k$) the consistency check fails $P_k$ must assume that all other parallel checks failed. Analogous modifications are made in Verify and Multiplication.

**Lemma 49.** *The modified Authenticate and Verify protocols have a total error probability of $\mathcal{O}(\frac{1}{|\mathbb{F}|}n^2)$ independent of the number of invocations.*

### 5.5.2 Broadcast

Although broadcast is only needed in Share, the total number of broadcast calls is in $\Theta(|\mathcal{Z}|)$. If [PW96] is used, the resulting overall error probability depends linearly on $|\mathcal{Z}|$. To avoid this problem, the number of broadcast calls must be reduced.

To reach this goal we use the fact that the Share protocol only has constantly many rounds. In each round a party $P_S$ must broadcast $\Theta(|\mathcal{Z}|)$ many messages of size $\mathcal{O}(\log|\mathbb{F}|)$. Instead of broadcasting these messages in parallel, $P_S$ sends their concatenation to the other parties, who then check that they received the same message. If an inconsistency is detected the protocol is repeated. To limit the number of repetitions we use the concept of dispute control from [BH06] which prevents the malicious parties from repetitive cheating. Dispute control is realized by a publicly known *dispute set* $\Gamma \subseteq \mathcal{P} \times \mathcal{P}$, a set of unordered pairs of parties. If $\{P_i, P_j\} \in \Gamma$ it means that there is a dispute between $P_i$ and $P_j$ and thus at least one of them is corrupted. Note that from $P_i$'s view all parties in $\{P_j | \{P_i, P_j\} \in \Gamma\}$ are malicious and thus he no longer trust them. At the beginning of the MPC protocol $\Gamma$ is empty.

**Protocol OptimisticBroadcast**$(\mathcal{P}, \mathcal{Z}, P_S, m)$

```
0: The party $P_S$ takes $m \in \{0,1\}^w$ as input.
1: $\forall \{P_i, P_S\} \notin \Gamma$ $P_S$ sends $m$ as $m_i$ to $P_i$.
2: $\forall \{P_i, P_j\} \notin \Gamma$ $P_i$ sends $m_i$ as $m_{ij}$ to $P_j$.
3: $\forall P_i$ if all received values are the same $P_i$ is happy, otherwise
   unhappy. $P_i$ broadcasts using [PW96] his happy bit.
```

```
4: If all parties are happy, each $P_i$ outputs the value he holds.
   Otherwise, an unhappy party $P_i$ (e.g. the one with the smallest
   index) broadcasts $j, j', z, b$ where $m_{ji}$ differs from $m_{j'i}$ at
   bit-position $z$ and $b$ is the bit of $m_{ji}$ at position $z$. Then $P_S, P_j, P_{j'}$
   broadcast their versions of the bit at position $z$. Using this
   information the parties localize a dispute between two parties of
   $\{P_i, P_S, P_j, P_{j'}\}$. Then the protocol is repeated with updated $\Gamma$.
```

**Lemma 50.** *The protocol* OptimisticBroadcast$(\mathcal{P}, \mathcal{Z}, P_S, m)$ *achieves the broadcast of a message* $m' \in \{0,1\}^w$. *The protocol communicates at most* $w \cdot \mathrm{poly}(n, \kappa)$ *bits and broadcasts at most* $\log w \cdot \mathrm{poly}(n, \kappa)$.

*Proof.* The properties of $\Gamma$ guarantee that honest parties will exchange in Step 2 their received messages from $P_S$. So if all honest parties are happy they all will output the same message $m'$. For an honest $P_S$ this also ensures that $m' = m$. If a party is unhappy, at least one party misbehaved. The actions taken in Step 4 then ensure that the honest parties will find at least one dispute. The protocol will terminate, as the number of repetition is limited by $n(n-1)$. As the broadcast of $z$ requires $\log w$ bits, the communication and broadcast complexities follow. □

For a message of length $\Theta(|\mathcal{Z}|)$ the above protocol only needs to broadcast $\log|\mathcal{Z}| \cdot \mathrm{poly}(n, \kappa)$ bits, hence the total number of broadcast calls per invocation of Share is reduced to $\log|\mathcal{Z}| \cdot \mathrm{poly}(n, \kappa)$.

**Lemma 51.** *The modified* Share *protocol communicates* $|\mathcal{C}||\mathcal{Z}| \cdot \mathrm{poly}(n, \kappa)$ *bits and broadcasts* $|\mathcal{C}| \log|\mathcal{Z}| \cdot \mathrm{poly}(n, \kappa)$ *bits.*

### 5.5.3 Summary

The combination of the above extension results in the following Lemma:

**Theorem 9.** *Let* $\mathcal{C}$ *be a circuit over* $\mathbb{F}$, *where* $|\mathbb{F}| \in \Omega(2^\kappa)$ *and* $\kappa$ *is a security parameter, and let* $\mathcal{Z}$ *be an adversary structure satisfying* $\mathcal{Q}^2(\mathcal{P}, \mathcal{Z})$, *then the modified protocol* MPC$(\mathcal{P}, \mathcal{Z}, \mathcal{C})$ $\mathcal{Z}$*-securely evaluates* $\mathcal{C}$ *with an error probability of* $2^{-\kappa}|\mathcal{C}| \cdot \mathrm{poly}(n, \kappa)$. *It communicates* $|\mathcal{C}||\mathcal{Z}| \cdot \mathrm{poly}(n, \kappa)$ *bits and broadcasts* $|\mathcal{C}| \log|\mathcal{Z}| \cdot \mathrm{poly}(n, \kappa)$ *bits. The number of rounds is* $\mathrm{poly}(n, \kappa) \cdot d$, *where* $d$ *denotes the multiplicative depth of* $C$.

*Proof.* Follows directly from Theorem 8 and Lemmas 49 and 51. □

By replacing broadcast with the simulated one from [PW96], one gets for $|\mathcal{Z}| \in \mathcal{O}(2^n)$ and $|\mathcal{C}| \in \text{poly}(\kappa)$ the following theorem.

**Theorem 10.** *Let $\mathcal{C}$ be a circuit over $\mathbb{F}$, where $|\mathbb{F}| \in \Omega(2^\kappa)$ and $\kappa$ is a security parameter, and let $\mathcal{Z}$ be an adversary structure satisfying $\mathcal{Q}^2(\mathcal{P}, \mathcal{Z})$, then $\mathsf{MPC}(\mathcal{P}, \mathcal{Z}, \mathcal{C})$ $\mathcal{Z}$-securely evaluates $\mathcal{C}$ with an error probability of $2^{-\kappa} \cdot \text{poly}(n, \kappa)$. It communicates $|\mathcal{Z}| \cdot \text{poly}(n, \kappa)$ bits.*

## 5.6 Lower Bound on the Efficiency

The following theorem states that there exists a family of circuits and $\mathcal{Q}^2$ adversary structures such that the length of unconditionally secure protocols tolerating these adversaries grows exponentially in the number of parties. This implies that the computational complexity of our protocol from the previous section is optimal, as there exists no protocol with a computational complexity in $o(|\mathcal{Z}|)$.

**Theorem 11.** *[Hir01] Let $\mathcal{C}$ be the circuit which takes inputs from $P_1$ and $P_2$ and outputs the product to $P_1$. Then there exists a family $\mathcal{Z}_2, \mathcal{Z}_3, \ldots$ of $\mathcal{Q}^2$ adversary structures for party sets $\mathcal{P}_2, \mathcal{P}_3, \ldots$ ($|\mathcal{P}_n| = n$) such that the length of the shortest unconditionally $\mathcal{Z}_n$-secure protocol for $\mathcal{C}$ grows exponentially in $n$.*

# Bibliography

[Bd90] Jurjen N. Bos and Bert den Boer. Detection of disrupters in the DC protocol. In Jean-Jacques Quisquater and Joos Vandewalle, editors, *Advances in Cryptology – EUROCRYPT'89*, volume 434 of *Lecture Notes in Computer Science*, pages 320–327. Springer, Heidelberg, April 1990.

[Bea91] Donald Beaver. Secure multiparty protocols and zero-knowledge proof systems tolerating a faulty minority. *Journal of Cryptology*, 4(2):75–122, 1991.

[Bea92] Donald Beaver. Efficient multiparty protocols using circuit randomization. In Joan Feigenbaum, editor, *Advances in Cryptology – CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, pages 420–432. Springer, Heidelberg, August 1992.

[BFH$^+$08] Zuzana Beerliová-Trubíniová, Matthias Fitzi, Martin Hirt, Ueli M. Maurer, and Vassilis Zikas. MPC vs. SFE: Perfect security in a unified corruption model. In Ran Canetti, editor, *TCC 2008: 5th Theory of Cryptography Conference*, volume 4948 of *Lecture Notes in Computer Science*, pages 231–250. Springer, Heidelberg, March 2008.

[BGP89] Piotr Berman, Juan A. Garay, and Kenneth J. Perry. Towards optimal distributed consensus (extended abstract). In *30th Annual Symposium on Foundations of Computer Science*, pages 410–415. IEEE Computer Society Press, October / November 1989.

[BGT13] Elette Boyle, Shafi Goldwasser, and Stefano Tessaro. Communication locality in secure multi-party computation - how to run sublinear algorithms in a distributed setting. In Amit Sahai, editor, *TCC 2013: 10th Theory of Cryptography Conference*, volume 7785 of *Lecture Notes in Computer Science*, pages 356–376. Springer, Heidelberg, March 2013.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *20th Annual ACM Symposium on Theory of Computing*, pages 1–10. ACM Press, May 1988.

[BH06] Zuzana Beerliová-Trubíniová and Martin Hirt. Efficient multiparty computation with dispute control. In Shai Halevi and Tal Rabin, editors, *TCC 2006: 3rd Theory of Cryptography Conference*, volume 3876 of *Lecture Notes in Computer Science*, pages 305–328. Springer, Heidelberg, March 2006.

[Bon98] Dan Boneh. *Algorithmic Number Theory: Third International Symposiun, ANTS-III Portland, Oregon, USA, June 21–25, 1998 Proceedings*, chapter The Decision Diffie-Hellman problem, pages 48–63. Springer, Heidelberg, 1998.

[Can98] Ran Canetti. Security and composition of multi-party cryptographic protocols. Cryptology ePrint Archive, Report 1998/018, 1998. `http://eprint.iacr.org/1998/018`.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science*, pages 136–145. IEEE Computer Society Press, October 2001.

[CCD88] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In *20th Annual ACM Symposium on Theory of Computing*, pages 11–19. ACM Press, May 1988.

[CCG$^+$15] Nishanth Chandran, Wutichai Chongchitmate, Juan A. Garay, Shafi Goldwasser, Rafail Ostrovsky, and Vassilis Zikas. The hidden graph model: Communication locality and optimal resiliency with adaptive faults. In Tim Roughgarden, editor, *ITCS 2015: 6th Innovations in Theoretical Computer Science*, pages 153–162. Association for Computing Machinery, January 2015.

[CDD$^+$99] Ronald Cramer, Ivan Damgård, Stefan Dziembowski, Martin Hirt, and Tal Rabin. Efficient multiparty computations secure against an adaptive adversary. In Jacques Stern, editor, *Advances in Cryptology – EUROCRYPT'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 311–326. Springer, Heidelberg, May 1999.

[CDN15] Ronald Cramer, Ivan Damgård, and Jesper Buus Nielsen. *Secure Multiparty Computation and Secret Sharing.* Cambridge University Press, 1st edition, 2015.

[CFF$^+$05] Jeffrey Considine, Matthias Fitzi, Matthew K. Franklin, Leonid A. Levin, Ueli M. Maurer, and David Metcalf. Byzantine agreement given partial broadcast. *Journal of Cryptology*, 18(3):191–217, July 2005.

[CGO15] Nishanth Chandran, Juan A. Garay, and Rafail Ostrovsky. Almost-everywhere secure computation with edge corruptions. *Journal of Cryptology*, 28(4):745–768, October 2015.

[Cha81] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84 – 90, February 1981.

[Cha88] David Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1(1):65–75, 1988.

[Cha03] David Chaum. Untraceable electronic mail, return addresses and digital pseudonyms. In Dimitris Gritzalis, editor, *Secure Electronic Voting*, volume 7 of *Advances in Information Security*, pages 211–219. Springer, Heidelberg, 2003.

[DDWY90] Danny Dolev, Cynthia Dwork, Orli Waarts, and Moti Yung. Perfectly secure message transmission. In *31st Annual Symposium on Foundations of Computer Science*, pages 36–45. IEEE Computer Society Press, October 1990.

[DJ01] Ivan Damgård and Mats Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In Kwangjo Kim, editor, *PKC 2001: 4th International Workshop on Theory and Practice in Public Key Cryptography*, volume 1992 of *Lecture Notes in Computer Science*, pages 119–136. Springer, Heidelberg, February 2001.

[ElG84] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology – CRYPTO'84*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer, Heidelberg, August 1984.

[FFGV07] Matthias Fitzi, Matthew K. Franklin, Juan A. Garay, and S. Harsha Vardhan. Towards optimal and efficient perfectly secure message transmission. In Salil P. Vadhan, editor, *TCC 2007: 4th Theory of Cryptography Conference*, volume 4392 of *Lecture Notes in Computer Science*, pages 311–322. Springer, Heidelberg, February 2007.

[FGH$^+$02] Matthias Fitzi, Daniel Gottesman, Martin Hirt, Thomas Holenstein, and Adam Smith. Detectable Byzantine agreement secure against faulty majorities. In Aleta Ricciardi, editor, *21st ACM Symposium Annual on Principles of Distributed Computing*, pages 118–126. Association for Computing Machinery, July 2002.

[FLM85] Michael J. Fischer, Nancy A. Lynch, and Michael Merritt. Easy impossibility proofs for distributed consensus problems. In Michael A. Malcolm and H. Raymond Strong, editors, *4th ACM Symposium Annual on Principles of Distributed Computing*, pages 59–70. Association for Computing Machinery, August 1985.

[FM98] Matthias Fitzi and Ueli Maurer. Efficient Byzantine agreement secure against general adversaries. In *DISC*, volume 1499 of *Lecture Notes in Computer Science*, pages 134–148. Springer, Heidelberg, September 1998.

[FM00] Matthias Fitzi and Ueli M. Maurer. From partial consistency to global broadcast. In *32nd Annual ACM Symposium on Theory of Computing*, pages 494–503. ACM Press, May 2000.

[GGOR14] Juan A. Garay, Clinton Givens, Rafail Ostrovsky, and Pavel Raykov. Fast and unconditionally secure anonymous channel. In Magnús M. Halldórsson and Shlomi Dolev, editors, *33rd ACM Symposium Annual on Principles of Distributed Computing*, pages 313–321. Association for Computing Machinery, July 2014.

[GJ04] Philippe Golle and Ari Juels. Dining cryptographers revisited. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 456–473. Springer, Heidelberg, May 2004.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th Annual ACM Symposium on Theory of Computing*, pages 218–229. ACM Press, May 1987.

[GO08] Juan A. Garay and Rafail Ostrovsky. Almost-everywhere secure computation. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 307–323. Springer, Heidelberg, April 2008.

[Gol01] Oded Goldreich. *The Foundations of Cryptography - Volume 1, Basic Techniques.* Cambridge University Press, 2001.

[GY89] Ronald L. Graham and Andrew Chi-Chih Yao. On the improbability of reaching Byzantine agreements (preliminary version). In *21st Annual ACM Symposium on Theory of Computing*, pages 467–478. ACM Press, May 1989.

[Hir01] Martin Hirt. *Multi-Party Computation: Efficient Protocols, General Adversaries, and Voting.* PhD thesis, ETH Zurich, September 2001. Reprint as vol. 3 of *ETH Series in Information Security and Cryptography*, ISBN 3-89649-747-2, Hartung-Gorre Verlag, Konstanz, 2001.

[HJ07] Markus Hinkelmann and Andreas Jakoby. Communications in unknown networks: Preserving the secret of topology. *Theoretical Computer Science*, 384(2-3):184–200, 2007.

[HM97] Martin Hirt and Ueli M. Maurer. Complete characterization of adversaries tolerable in secure multi-party computation (extended abstract). In James E. Burns and Hagit Attiya, editors, *16th ACM Symposium Annual on Principles of Distributed Computing*, pages 25–34. Association for Computing Machinery, August 1997.

[HM00] Martin Hirt and Ueli M. Maurer. Player simulation and general adversary structures in perfect multiparty computation. *Journal of Cryptology*, 13(1):31–60, 2000. Extended abstract in *Proc. 16th of ACM PODC '97*.

[HMR14] Martin Hirt, Ueli Maurer, and Pavel Raykov. Broadcast amplification. In Yehuda Lindell, editor, *TCC 2014: 11th Theory of Cryptography Conference*, volume 8349 of *Lecture Notes in Computer Science*, pages 419–439. Springer, Heidelberg, February 2014.

[HMTZ16] Martin Hirt, Ueli Maurer, Daniel Tschudi, and Vassilis Zikas. Network-hiding communication and applications to multiparty protocols. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part II*, volume 9815 of *Lecture Notes in Computer Science*, pages 335–365. Springer, Heidelberg, August 2016.

[HMZ08] Martin Hirt, Ueli M. Maurer, and Vassilis Zikas. MPC vs. SFE: Unconditional and computational security. In Josef Pieprzyk, editor, *Advances in Cryptology – ASIACRYPT 2008*, volume 5350 of *Lecture Notes in Computer Science*, pages 1–18. Springer, Heidelberg, December 2008.

[HT13] Martin Hirt and Daniel Tschudi. Efficient general-adversary multi-party computation. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology – ASIACRYPT 2013, Part II*, volume 8270 of *Lecture Notes in Computer Science*, pages 181–200. Springer, Heidelberg, December 2013.

[JMS12] Alexander Jaffe, Thomas Moscibroda, and Siddhartha Sen. On the price of equivocation in Byzantine agreement. In Darek Kowalski and Alessandro Panconesi, editors, *31st ACM Symposium Annual on Principles of Distributed Computing*, pages 309–318. Association for Computing Machinery, July 2012.

[KNS16] Anna Krasnova, Moritz Neikes, and Peter Schwabe. Footprint scheduling for dining-cryptographer networks. In Jens Grossklags and Bart Preneel, editors, *FC 2016: 20th International Conference on Financial Cryptography and Data Security*, volume 9603 of *Lecture Notes in Computer Science*, pages 385–402. Springer, Heidelberg, February 2016.

[KS08] Kaoru Kurosawa and Kazuhiro Suzuki. Truly efficient 2-round perfectly secure message transmission scheme. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 324–340. Springer, Heidelberg, April 2008.

[KS10] Valerie King and Jared Saia. Breaking the $O(n^2)$ bit barrier: scalable Byzantine agreement with an adaptive adversary. In Andréa W. Richa and Rachid Guerraoui, editors, *29th ACM Symposium Annual on Principles of Distributed Computing*, pages 420–429. Association for Computing Machinery, July 2010.

[KSSV06a] Valerie King, Jared Saia, Vishal Sanwalani, and Erik Vee. Scalable leader election. In *17th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 990–999. ACM-SIAM, January 2006.

[KSSV06b] Valerie King, Jared Saia, Vishal Sanwalani, and Erik Vee. Towards secure and scalable computation in peer-to-peer networks. In *47th Annual Symposium on Foundations of Computer Science*, pages 87–98. IEEE Computer Society Press, October 2006.

[KY84] Anna Rochelle Karlin and Andrew Chi-Chih Yao. Probabilistic lower bounds for the Byzantine generals problem. unpublished manuscript, 1984.

[Lam83] Leslie Lamport. The weak Byzantine generals problem. *Journal of the ACM*, 30(3):668–676, July 1983.

[LMT16] Julian Loss, Ueli Maurer, and Daniel Tschudi. Hierarchy of three-party consistency specifications. In *2016 IEEE International Symposium on Information Theory (ISIT)*, pages 3048–3052. IEEE, July 2016.

[LMT17] Julian Loss, Ueli Maurer, and Daniel Tschudi. Strong separations between broadcast and authenticated channels. unpublished manuscript, 2017.

[LSP82] Leslie Lamport, Robert Shostak, and Marshall Pease. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3):382–401, July 1982.

[Mau03] Ueli M. Maurer. Secure multi-party computation made simple (invited talk). In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *SCN 02: 3rd International Conference on Security in Communication Networks*, volume 2576 of *Lecture Notes in Computer Science*, pages 14–28. Springer, Heidelberg, September 2003.

[Mau04] Ueli Maurer. Towards a theory of consistency primitives. In Rachid Guerraoui, editor, *International Symposium on Distributed Computing — DISC 2004*, volume 3274 of *Lecture Notes in Computer Science*, pages 379–389. Springer, Heidelberg, October 2004.

[Mau11] Ueli Maurer. Constructive cryptography – a new paradigm for security definitions and proofs. In S. Moedersheim and C. Palamidessi, editors, *Theory of Security and Applications*

*(TOSCA 2011)*, volume 6993 of *Lecture Notes in Computer Science*, pages 33–56. Springer, Heidelberg, April 2011.

[MOR15] Tal Moran, Ilan Orlov, and Silas Richelson. Topology-hiding computation. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part I*, volume 9014 of *Lecture Notes in Computer Science*, pages 159–181. Springer, Heidelberg, March 2015.

[MR11] Ueli Maurer and Renato Renner. Abstract cryptography. In Bernard Chazelle, editor, *ICS 2011: 2nd Innovations in Computer Science*, pages 1–21. Tsinghua University Press, January 2011.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *Advances in Cryptology – EUROCRYPT'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer, Heidelberg, May 1999.

[Pas04] Rafael Pass. Bounded-concurrent secure multi-party computation with a dishonest majority. In László Babai, editor, *36th Annual ACM Symposium on Theory of Computing*, pages 232–241. ACM Press, June 2004.

[PSR03] B. Prabhu, K. Srinathan, and C. Pandu Rangan. Trading players for efficiency in unconditional multiparty computation. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *SCN 02: 3rd International Conference on Security in Communication Networks*, volume 2576 of *Lecture Notes in Computer Science*, pages 342–353. Springer, Heidelberg, September 2003.

[PW96] Birgit Pfitzmann and Michael Waidner. Information-theoretic pseudosignatures and Byzantine agreement for $t \geq n/3$. Research report, IBM Research, 1996.

[Ray15] Pavel Raykov. Broadcast from minicast secure against general adversaries. In Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *ICALP 2015:*

*42nd International Colloquium on Automata, Languages and Programming, Part II*, volume 9135 of *Lecture Notes in Computer Science*, pages 701–712. Springer, Heidelberg, July 2015.

[RB89] Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In *21st Annual ACM Symposium on Theory of Computing*, pages 73–85. ACM Press, May 1989.

[RMS$^+$04] D. V. S. Ravikant, Venkitasubramaniam Muthuramakrishnan, V. Srikanth, K. Srinathan, and C. Pandu Rangan. On Byzantine agreement over (2,3)-uniform hypergraphs. In Rachid Guerraoui, editor, *International Symposium on Distributed Computing — DISC 2004*, volume 3274 of *Lecture Notes in Computer Science*, pages 450–464. Springer, Heidelberg, October 2004.

[RR98] Michael K. Reiter and Aviel D. Rubin. Crowds: Anonymity for web transactions. *ACM Trans. Inf. Syst. Secur.*, 1(1):66–92, November 1998.

[SGR97] Paul F. Syverson, David M. Goldschlag, and Michael G. Reed. Anonymous connections and onion routing. In *1997 IEEE Symposium on Security and Privacy, May 4-7, 1997, Oakland, CA, USA*, pages 44–54. IEEE Computer Society, 1997.

[SNR04] K. Srinathan, Arvind Narayanan, and C. Pandu Rangan. Optimal perfectly secure message transmission. In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 545–561. Springer, Heidelberg, August 2004.

[UKBM11] Johan Ugander, Brian Karrer, Lars Backstrom, and Cameron Marlow. The anatomy of the facebook social graph. *CoRR*, abs/1111.4503, 2011.

[Yao82] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science*, pages 160–164. IEEE Computer Society Press, November 1982.

# Curriculum Vitae

**Daniel Tschudi**
Citizen of Glarus GL and Zürich ZH, Switzerland.
Born on 15 May 1988, in Horgen ZH, Switzerland.

**Doctoral Studies** *09/2012 - 01/2018*

ETH Zürich, Department of Computer Science
Thesis: Selected Topics in Secure Multi-Party Computation
Advisor: Prof. Dr Ueli Maurer
Co-examiners: Prof. Dr. Jesper Buus Nielsen, Dr. Martin Hirt
Degree: Doctor of Sciences (Dr. sc. ETH)

**Undergraduate Studies** *09/2007 - 09/2012*

ETH Zürich, Department of Mathematics
Thesis: Complexity of Multi-Party Computation Secure against General Adversaries
Degree: Master of Science ETH in Mathematics

**High School** *08/2001 - 08/2012*

School: Kantonsschule Freudenberg, Zürich, Switzerland